Adaptive Game Input Using Knowledge of Player Capability: Designing for Individuals with Different Abilities

A Thesis

Submitted to the Faculty

of

Drexel University

by

Robert C. Gray

in partial fulfillment of the

requirements for the degree

of

Master of Science in Digital Media

March 2018

For my wife, Megan, whose example inspired me to pursue research

Acknowledgments

I would like to thank my advisor Dr. Paul Diefenbach, whose guidance and encouragement has empowered me to pursue this lifelong ambition. Your deep insight and tireless commitment toward helping me formalize and synthesize this idea has been instrumental in shaping this work.

I am also grateful for my advisor Dr. Jichen Zhu, whose mentorship and instruction continues to shape my worldview and perspectives as a researcher. Your passion for our field and dedication to your craft are an inspiration to me, and I am thankful for how you have expanded my understanding of the practice of research and the philosophy of science.

I would like to express my gratitude to Dr. Michael Wagner for his support and advice as a member of my thesis committee, as well as Dr. Glen Muschio and Dr. Stefan Rank for their excellent instruction and guidance throughout my journey.

My return to academia would not have been possible without the help of Timothy J. Day, a great friend and colleague whose assistance and selflessness helped me through challenges these past several years both in life and in study. For the same reason, I extend my gratitude to Dr. Frank Lee, my first introduction to Drexel University, who gave me a spot on his team and his projects and guided me toward this path.

And finally, I would like to thank my family, Dr. Megan Gray along with Jackson and Chase, who endured late nights and many other sacrifices to enable me to pursue this dream. I hope to make you all proud.

Table of Contents

List of Tables

List of Figures

Abstract

Adaptive Game Input Using Knowledge of Player Capability: Designing for Individuals with Different Abilities

Robert C. Gray
Advisors: Paul Diefenbach, Ph.D., Jichen Zhu, Ph.D.

The application of video games has been shown to be valuable in medical interventions, such as the use of Active Video Games (AVGs) in physical therapy. Because patients requiring physical therapy present with both highly variable physical capabilities and unique therapeutic goals, developers of rehabilitation intervention games face the challenge of creating flexible games that they can individualize to each player's particular needs.

This thesis proposes an approach to this problem by identifying and addressing two issues concerning therapy AVG game design. First, regarding the difficulties of individualizing software, a particular complication in the development of AVGs for therapy is the increased complexity of writing input routines based on human body motion, which provides a much larger and more complex domain than traditional, discrete-input game controllers.

Second, the primary difficulty in individualizing a therapy game experience to an individual player is that developers must program software with static routines that cannot be modified once compiled and released. Overcoming this aspect of software development is a prime concern that adaptive games research aims to address.

The *System for Unified Kinematic Input* (SUKI) is a software library that addresses both of these concerns. SUKI enables games to adapt to players' specific therapeutic goals by mapping arbitrary human body movement input to game mechanics at runtime, allowing user-defined body motions to drive gameplay without requiring any change to the software. Additionally, the SUKI library implements a

dynamic profile system that alters the game's configuration based on known physical capabilities of the player and updates this profile based on the player's demonstrated ability during play.

Within the context of the study of adaptive games, the following research presents the details of this approach and demonstrates the versatility and extensibility that it can provide in adapting AVG games to meet individual player needs.

1. INTRODUCTION

The use of video games in medicine, and particularly the role of Active Video Games (AVGs) in physical therapy interventions, holds great promise for the 38 million individuals in the United States alone that have severe physical, cognitive, and sensory disabilities or chronic conditions that result in functional limitations [12]. Recent studies have shown that leveraging games in movement therapy can lead to patient-level outcomes that match or even outperform traditional therapy [50].

Whether playing traditional games for entertainment or those designed for therapy, disabilities can impact player participation and enjoyment in varied ways. Such players may present with limited strength, flexibility, endurance, or control in movement functions required for input in a motion-based game. Beyond the physical aspects addressed by the game system in this research, other player traits such as working memory capacity, spatial reasoning, and even more abstract qualities like player temperament may be valuable focal points around which games can adapt during play. Most importantly, if developers can empower games to better adapt to the unique capabilities of their players, they may be able to provide better experiences beyond this specific player population and to video game players in general.

1.1 Games as Therapy

Physical therapy is the central component for treatment systems in cases of chronic neurological conditions such as cerebral palsy [9] and Parkinson's disease [48], where this care is traditionally administered under the supervision of a physical therapist. These therapy sessions typically focus on specific exercises that promote motor activity, which have been shown to prevent secondary impairments, facilitate neural recovery, improve cognitive performance, and lead to improved mental and physical health [21]. Therapists often prescribe a daily home regimen of exercises to maintain strength and mobility between sessions, which is where AVGs have significant opportunity to improve therapeutic outcomes.

Using motion-based input systems, AVGs are games that interpret movement in the player's body and compare it to a motion expectation or apply it toward in-game objectives. Cerebral palsy (CP) is often selected as a targeted disability in therapy games research due to its relatively high occurrence and symptom diversity. However, research has shown this method to be successful in treating an array of disabilities including Parkinson's disease [32, 63], cystic fibrosis [57, 38], multiple sclerosis [50, 35], burn rehabilitation [59], and others [42, 66].

The ability to individualize care and provide specific goals can have a large effect on the quality and effectiveness of physical therapy. For example, because some disabilities like CP can vary widely in how they present symptomatically (concerning both physical and cognitive impairment), there is not yet any specific therapy that can apply broadly to every patient; therefore, management of patients requires frequent readjustment [9]. In a small-sample study, Frankie et al. [30] also found individually tailored physical therapy programs to yield more favorable results when treating youth with CP. In the case of similar neuromotor disabilities, patients with traumatic brain injury have shown significantly better improvements on manual dexterity tests when they are given specific, difficult goals instead of general guidance [33].

1.2 Motion Based Input

From this, we understand that a key component of improving therapy games is to empower them to provide more specialized and individualized experiences for their users. A key element of this individualization is the way in which the game system interprets body motion as input and applies the player's intent toward game objectives. Traditionally, specialized software routines accomplish this by watching for pre-determined and anticipated movements (sometimes in the form of *gestures*) and updating specific gameplay elements according to the detected movement.

Though straightforward and efficient, the specific mapping of gestures to gameplay elements limits a single game's potential to a specific therapy outcome and does not effectively consider variance among patient capabilities. Games that use traditional button-based input devices can profit from the convenience of easily remapping their keys within and among devices (such as with a keyboard, mouse, or game controller). Movement-based AVGs could greatly benefit from a similar affordance; however, the domain of analog human movement is vastly larger than that of a discrete key-based device.

1.3 Adaptive Games

Video games, more so than any other form of media to date, offer a unique affordance to their audience in that they inherently possess an ability to alter themselves in real time based on the knowledge of the viewer. Studies in the field of adaptive games attempt to improve the play experience by changing the game environment and mechanics in a way that is expected to align preferably with the individual player's preferences. However, current approaches focus primarily on high-level performance metrics and the preferences and predilections of a player concerning games without considering other important aspects (e.g., capabilities, behavioral tendencies, and tolerances) of the player that may also greatly influence the quality of interaction.

Key to our investigation are the physical capabilities of the player and their individual therapeutic goals as prescribed by their therapist. Though various adaptive games studies base their adaptation on player performance, in this research we are concerned with how player capability might serve as a useful source of information for an adaptive game system. To this end, we present a system that enables therapists to individualize a single game toward a broad range of therapy goals among patients of varying physical limitations without any need to change the game's programming.

With this work we propose the design of a specialized software architecture called the *System for Unified Kinematic Input* (SUKI) for decoupling the software tasks of interpreting of body movement and managing game-specific input signals. Further, we demonstrate how a player model unique to each patient can inform this abstraction and provide an avenue for adaptive game design. In section two we will present an overview of the problem domain and discuss the need for this kind of solution in physical rehabilitation therapy AVGs. In section three we will review the current literature on player models and adaptive games and discuss how we can leverage the dynamic qualities of computer software along with knowledge of the player to meet these needs. Section four will present an overview of the SUKI system as a response to the needs and opportunities presented in previous sections and discuss its requirements, architecture, and feature implementation. Section five will evaluate the solution by exploring three case studies, where we will build a therapy AVG with SUKI, use SUKI to adapt a game to a new therapy goal and patient population, and finally retrofit an existing AVG with SUKI to expand its therapeutic potential. Section six will present a discussion of our findings and suggestions for future work.

2. BACKGROUND

Addressing this problem domain requires that we examine how researchers and therapists have adopted body motion games for therapy and the nature of the challenges that developers face when creating these games. The following is a brief overview of the history and progress of AVG technology and a discussion of the solutions available in the industry today. We also examine the special technical considerations developers face when creating active video games, particularly for the purpose of physical therapy. Finally, we discuss the primary need for accessibility and individualization in creating effective AVG therapy experiences, where insight from adaptive game theory may further enhance the effectiveness of these therapy tools.

2.1 Active Video Games

The Active Video Game is a category of "games that require players' body movements for game play" [62]. Beyond the prerequisite use of a *kinematic* (human body movement) sensor for player input, the term "active" usually implies that the game is designed with some form of exercise goal in mind. Early examples of AVGs for consumers include the Foot Craz for the Atari 2600 and the Family Trainer and Power Pad peripherals for the Nintendo Entertainment System in the mid-1980s [8]. Around the same time, other projects like Autodesk's *Virtual Racquetball* [16] and RacerMate's *CompuTrainer* [11] began adding gaming elements to existing exercise equipment.

These early explorations were often limited to resolving complex body movement to a single value that represented exercise output, where this value when then leveraged as simplified player input in the game environment. Over the subsequent decades, more sophisticated systems like the Nintendo Wii and its Wii Fit exercise peripheral in 2007 [34] included balance boards and hand-held controllers with accelerometers that could detect user motion at a high frequency with multiple degrees of freedom. By enable more complex and multidimensional measurements of body movement, modern AVGs can take advantage of a higher level of player expression and incorporate more aspects of the player's movement in the game play.

Following the November 2010 launch of Microsoft's first-generation Kinect to pair with its popular Xbox 360 console system, the consumer-affordable body motion sensor device soon became the fastest-selling peripheral of all time [78]. A series of home exercise and dance training games followed, along with the second version of the sensor in late 2013 alongside Microsoft's Xbox One console. Though Microsoft eventually discontinued the product line as they transitioned the technology into the Microsoft Hololens device [75], ongoing academic projects and hobbyist communities continue to explore the potential for consumer affordable kinematic sensors in medical and industrial applications.

The sensor technology developed by PrimeSense [25] and implemented in the Kinect inspired a new industry of affordable, consumer-level kinematic sensors in the years to follow, including the Orbbec Astra [58] and Leap Motion Controller [74], both of which are employed in research exploring the potential of game-assisted physical therapy (Figures 2.1 and 2.2). In particular, the physical rehabilitation research and practitioner communities have found the devices to be appealing in their promise because they provide the necessary link for combining physical motion with the motivational power of video games.

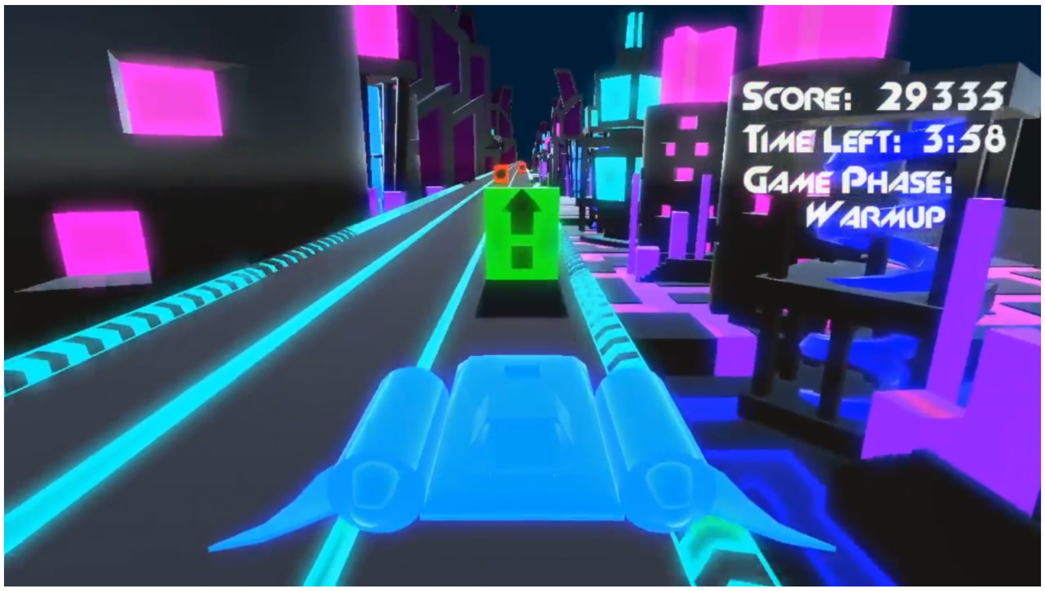


Figure 2.1: Research AVG *Citadel* uses the Microsoft Kinect and Orbbec Astra for upper-arm rehabilitation. Players in a seated or standing position can raise either their left or right arm over their head to move the ship among three lanes, collecting powerups and dodging obstacles.

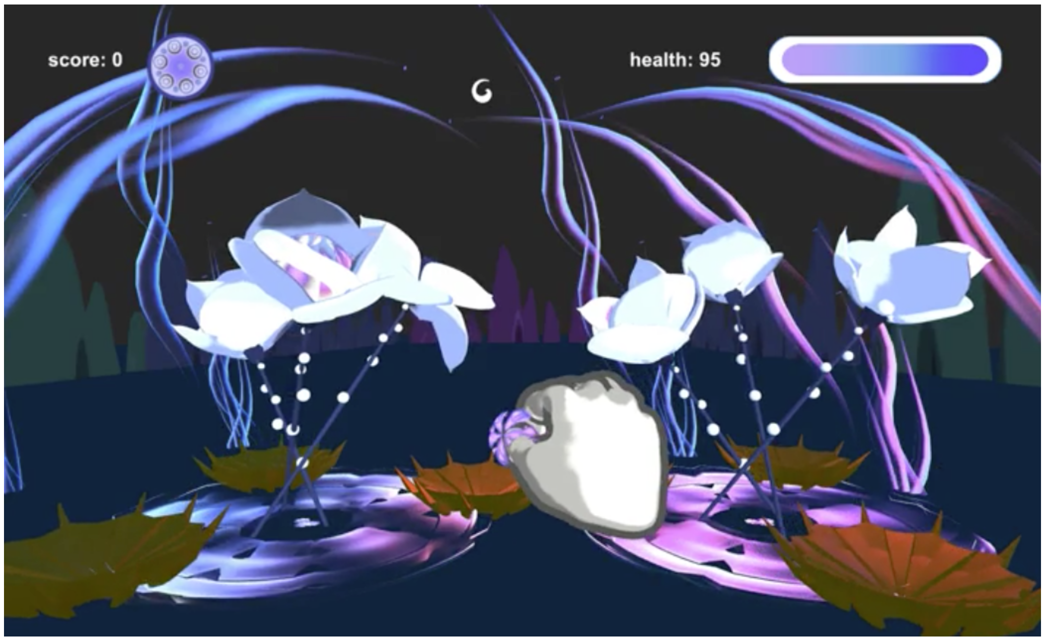


Figure 2.2: Research AVG *Sense Thief* uses the Leap Motion Controller to train fine motor skills. With a virtual hand matching the movements of the patient's real hand, players grab orbs from floating lily pads and place them in the rotating flowers.

2.2 Commercial Exergames

The advancement in sophistication of movement sensors has led to a commensurate advancement in the affordances of AVG experiences. Evolving from a single exertion metric to complex body motion input has reduced the necessity of external equipment to guide motion and resolve the metric (e.g., a physical exercise bike that governs a precise leg motion and submits the resulting RPM metric); similarly, this evolution has enabled a more complicated relationship between movement and gameplay, leading to an industry of commercial *exergames* (a portmanteau of "exercise" and "games") for both entertainment and therapy.

Though the Nintendo Wii has been a favored option in many therapy research projects [3, 10, 26, 45, 67], many of the studies are limited to commercial off-the-shelf (COTS) games with the hope that they align with the research and therapy goals. Though feasibility test results are often positive, they are not necessarily optimal; some studies note that leveraging COTS games toward therapy can encourage noneffective and undesired player movements, provide unfair or detrimental negative feedback, and fail to expose the customization parameters necessary for the target patient populations [1, 51, 64]. Even if an obvious and trivial change could assist therapy goals, researchers employing COTS software are not given extensive opportunity for customization. Due to the proprietary qualities of these games and the historically more restrictive nature of development for consoles (i.e. requiring developer registration and specialized hardware), end-user development for more specific purposes is often difficult even with requisite programming skill. In contrast, peripheral sensors for common personal computer (PC) software (e.g. Kinect, Leap Motion) have wider accessibility for focused research purposes.

For example, the Jintronix rehabilitation system [44] offers software based on the Kinect that includes both therapy games and clinically verified assessments. The software is proprietary and cannot be updated by end users, but it is still flexible in allowing a high degree of customization, including real-time interaction between the patient and a therapist operating at a separate console. However, the therapy games offered by Jintronix, though derived from traditional therapy routines, do not provide a way to map multiple motions to the same game experience.

Similarly, the VAST Rehab software suite [28] allows for real-time provider interaction with the game environment, including increasing or decreasing difficulty (e.g., number of enemies). Beyond many of its competitors, VAST Rehab also allows for the mapping of different motions to the same game, offering a selection of "control modes" in the form of movement patterns pre-written by the development team. To support this extended selection of control modes, it also supports multiple devices, including the Kinect, Leap Motion Controller, and specialized third-party medical devices. However, though it allows the creation of therapist-defined packages of combined control modes, end users are still limited to the activities defined by the developers and are not empowered to link gameplay objectives to arbitrary patient movements in the case where a new exercise is necessary.

Though not released as a commercial product, a notable tool for body movement control of a PC is the Flexible Action and Articulated Skeleton Toolkit (FAAST) project [70]. The FAAST project provides an input emulator for the player's operating system (OS) that generates traditional keyboard and mouse signals for the foreground application based on gestures it detects via a PrimeSense sensor. An advantage of this approach is its program-agnostic ability to interact with arbitrary software running on that OS through this emulation. However, like the VAST Rehab system, the gestures it can detect are limited to a pre-built "action lexicon" without an ability for end users to descriptively define new gestures. The project proposes to eventually add machine learning techniques for manually training custom gestures, but it does not address the fact that not all therapeutic exercises are gesture-based. Finally, the body movements detected by FAAST are reduced to emulated key presses and mouse input, limiting the expressiveness of the input system and requiring OS mediation, rather than being consumed directly by the game software.

2.3 Therapy AVG Programming

The creation of AVG games, in particular for therapy purposes, carries several unique challenges not often encountered in typical game development. Beyond additional concerns in design and testing, developing for kinematic sensors can be more complicated than developing for typical button-based control devices.

When a game is intended to use a kinematic sensor as an input device, the most straightforward path to implementing the game is to integrate the Software Development Kit (SDK) made available by the sensor's creator (e.g., Microsoft). Using this SDK, developers can search for specific body movement gestures or track targeted body parts as reported by the sensor and monitor for predetermined motions that they can then use to drive gameplay.

Unfortunately, this practice often solidifies the purpose of the game and thereby defines the extent of its use; a player whose required therapy regimen does not involve that specific movement will not need to play the game, nor is that patient likely to see expected therapeutic outcomes from playing it. Changes in the targeted body part or motion require the developer to rewrite the input routine and recompile (rebuild from source code) the game, after which they must redistribute it to the players. Even if source code were available to the player or therapist, specific knowledge would be required for these end users to make the modifications themselves.

Additionally, where many therapeutic goals might be achieved through body motions detectable by a single sensor, the game's therapeutic application is further enhanced by support for multiple sensor types. A minority of existing solutions, such as the aforementioned VAST Rehab software, do provide support for multiple sensors and select exercise equipment. For example, one patient's therapy task can be assigned a full body control mode detected by the Kinect, while another can be assigned a fine motor control mode using the Leap Motion Controller for the same game. Though support for multiple devices will likely always need to be explicitly provided by the framework developer, such solutions greatly increase the range of movements that can be assigned to gameplay. Though examination of multiple device support is beyond the scope of this research, SUKI currently supports the Microsoft Kinect and Orbbec Astra and has been designed to easily extend to support additional devices.

Games are often time-consuming and expensive to create, so any practice that developers can employ to make their games applicable to broader forms of therapy enable them to render more return on their investment. Similarly, games can be costly to purchase, so therapists are eager to find more ways to re-purpose existing games for multiple patients and a broader set of therapeutic goals. Finally, a given patient may like the gameplay experience of some games more than others, and therefore players themselves would also likely benefit from an ability to adapt the games of their choosing toward the particular therapies in which they are engaged.

2.4 Accessibility and Individualization

Beyond the need for more flexible input in the design of AVGs, accessibility and player capability are primary concerns to AVG developers. More recently, the games industry has begun to include consideration for player accessibility among their offerings. However, aside from merely providing user interface (UI) options to support a broader range of players (e.g., colorblind mode), there are scenarios in which the gameplay itself should adapt to accommodate individuals, especially when the game's purpose is to provide therapy for those players.

To address this, physical therapy researchers are already investigating games that can be adjusted for player capabilities in the physical space, calibrating gameplay expectations to known player physical limits. For instance, Paul J. Diefenbach et al. have designed systems that can take into account a player's reaching distance and joint flexion to prevent gameplay objectives that would be impossible for them [22]. However, there is a reason to believe that other factors might similarly improve player enjoyment and engagement (and by extension improve AVG therapy effectiveness) if game developers properly accounted for them.

The therapy AVG software offered by Doctor Kinetic [24] and Jintronix [44] include an automated patient calibration phase prior to exercise, but this must be performed manually on a regular basis in order for the system to adjust to changes in the player over time. Additionally, the calibration must be performed outside of a regular session prior to play; though the system can adjust itself to the patient's needs, it does not do so in real time during game play. In this way the software is highly configurable, but not necessarily *adaptive*.

Physical therapy is employed as an intervention for many types of conditions and disabilities [32, 38, 50, 59], and the patients targeted by these interventions present with highly varied symptoms and needs [17, 42, 50]. We propose that the application of serious game theory, particularly the techniques presented in the field of adaptive games research, is specially positioned to assist in implementing optimal physical therapy strategies. With such an approach, it is possible that methods previously employed to improve gameplay experience can instead work to tailor therapy not only to each player but to each player as they themselves change over time [55]. Though evaluating clinical efficacy is beyond the scope of this research, a mechanism like SUKI is a needed component for researchers to conduct such investigations.

To this end, the following section explores fundamental theory behind adaptive game techniques, including player modeling, dynamic difficulty adjustment, and a proposed taxonomy for approaches to game adaptation. Through this discussion, we will examine how a solution like SUKI can contribute to this field and potentially provide a method for increasing accessibility and individualization in therapy AVGs.

3. RELATED WORK

A principle approach to maximizing enjoyment within a video game is to take advantage of the fact that games, unlike most other forms of media, provide experiences that can adapt to new information in real time. Due to the natural involvement of a computer in video games and the algorithmic nature of their interactive systems, the game can "experience" the player at the same time the player is experiencing the game. As Janet H. Murray explains, the addition of a computer element demands a participatory response, and as a result, the relationship between the audience and digital media artifacts should be viewed as reciprocal [56].

One could argue that this reciprocal relationship is continually realized merely in the constant adjustment of the game environment based on player controls. As a player directs an avatar through a virtual environment and manipulates artifacts within the game, the software responds by accepting these instructions, altering state, and offering affordances. Though explicit player instruction to the environment is a valuable basis for adaptation [13], there remain opportunities for a game to adapt itself at a broader and more fundamental level based on an understanding of the player aside from the player's conscious action. Games that attempt to infer information about the player behind the controls and dynamically adjust to suit that player are known as *adaptive games*.

Adaptive games can be informed by a player profile reflecting one of the several theoretical models, or they can infer characteristics of the individual player through a deeper analysis of that player's in-game behavior. Because the general goal of an adaptive game is to maximize player enjoyment and engagement, we will first discuss the concept of *flow* [20], followed by a discussion of player modeling in games. Following this, we will then discuss a typical implementation of adaptive gaming called Dynamic Difficulty Adjustment (DDA) followed by an overview of the *bases* for adaptation commonly seen in adaptive games research.

3.1 Psychological Flow

Psychologist Mihaly Csikszentmihalyi [20] presented the concept of psychological flow in his landmark publication exploring how individuals achieve a sense of enjoyment in various activities. As Csikszentmihalyi explains, fulfillment is most likely achieved when a person engages in an activity that provides a challenge appropriate to their ability. When the challenge of an activity exceeds one's ability, the individual might become frustrated; however, approaching a challenge that does not meet one's ability could cause boredom. According to Tracy Fullerton, this concept for games implies that, to evoke happiness through continual achievement, the difficulty of an experience should increase or decrease along with the player's ability [31].

Not limited to merely a ratio of ability to challenge, the concept of flow also addresses other aspects of activities. These include the suitability of a task toward an individual's existing skillset, the clarity and perceived achievability of goals, the quality of incremental feedback yielded by the interaction, and the level of agency the activity provides [20]. Maximizing the player's sense of flow is key to maximizing the incentive for the player to engage the game and enjoyment derived while playing [31].

3.2 Player Modeling

Adaptive games often rely on player modeling, in which the game either draws from an existing set of data on the player or builds an internal model of the player at runtime or between play sessions [27] by observing behavior and discerning motive and player preference. Player models aim to capture a definition of a player's traits and the underlying motivations of a player [37].

Conventional approaches to maximizing player enjoyment in a game, particularly from the perspective of one designing games or offering games as a recommendation, is to evaluate characteristics of the player and attempt to pair them with games that exhibit qualities known to correlate well with those profile features. As exemplified below, research along these lines aims to improve our models for defining player profiles and establish stronger correlative evidence between an individual's attributes and notable aspects of gameplay.

A seminal work in the investigation of player types was presented by Richard Bartle [6], in which he examined the motivations of players in Multi-User Dungeon (MUD) games. Though primarily drawn from qualitative insight, his findings have served as a fundamental reference in the field of game design often referred to as the "Bartle Taxonomy."

Interviewing and observing the written communication of a few dozen experienced players of the genre, Bartle derived a model in which he separated gamers into four distinct categories, each driven by a particular game activity. Achievers are players focused on succeeding within the game's parameters and definitions for success and pride themselves in the commendations granted to them (publicly or privately) by the game. Explorers are players enticed by the promise of new knowledge within and about the game; they find enjoyment in possessing a deeper understanding of the game's breadth and depth, both concerning the game environment and the technology that implements it. Socializers leverage the game as a medium through which they can make interpersonal connections, and they assess their value in terms of the relationships and political stature they hold among the other players in the game. Killers delight in their ability to impose on others players and influence the gameplay experience of others (positively or negatively) with the affordances that a virtual environment can provide.

As video games have evolved, so has the nature of the models we use to describe player motivations. More recent studies into gamer motivation and identity by Nick Yee [76, 77] have extended the work presented by Bartle. As part of his *Daedalus* Project, Yee offered a more quantitative approach to validating his intuitive conclusions via data collected from thousands of massively multiplayer online role-playing game (MMORPG) players [76]. In doing so, Yee refined the model offered by Bartle based on an exploratory factor analysis.

Aroutis N. Foster examined player models in the context of education science, where analysis of children playing a simulation strategy game suggested a dual-axis system for modeling learning styles [29]. In this model, he evaluates player learners for the value they place in achieving in-game goals (Goal Seekers) or experiencing and mastering the content (Explorers), which he then positions on an axis orthogonal to their tendencies toward introversion or extroversion. Goal Seekers with lower sociability are Achievers who pursue personal successes, while those with higher sociability classify as Competitors driving for more visible successes compared to other players. Explorers with less sociability are Comprehensive Explorers who desire a breadth of game knowledge, while those with more sociability are Localized Explorers in pursuit of depth in specialized knowledge. Studies like those by Justin H. Patterson have explored the application and prediction of Foster's model through examination of gameplay metrics [61].

3.2.1 Non-static Player Models

Though researchers often approach player models with the assumption that they convey constant and static characteristics regarding the player, recent research by Josep Valls-Vargas et al. suggests that a player's model may change or evolve in the course of play [72]. Proposing a more flexible framework based on the Foster model that evaluates player behavior within smaller temporal segments of a larger play session, the study uses sequential machine learning techniques to predict these shifts. One challenge presented and explored in this approach is the level of granularity chosen in the analysis, where evaluating time segments that are too small or too large reduce the accuracy of the analysis. Exploring the idea that player models should not be viewed as static but possibly migratory as the player increases exposure to the media and themselves change over time is a departure from earlier studies; even so, it is of particular interest to our investigation of an activity intentionally designed to promote change in the player.

3.3 Dynamic Difficulty Adjustment

Catering directly to the concept of flow, one of the most common approaches to adaptive games research is exploring techniques for determining optimal game difficulty. Jenova Chen [15] explored in his master's thesis the use of a Dynamic Difficulty Adjustment (DDA) system in an attempt to apply the concept of flow toward a more engaging game experience. Chen's work extends a line of inquiry that includes the research of Robin Hunicke and Vernell Chapman before him, in which they employed a DDA system to manipulate the economy of a game to grant or restrict resources based on the player's performance [39].

More recent studies have explored the difficulties of managing DDA systems in multiplayer competitive games [60] and how psychological and sociological models can help inform the game on how to offer the most compelling types of challenge [36]. Justin Alexander et al. examined the impact of DDA systems on different player types (namely, casual vs. experienced gamers) to discover several differences between them. Among their findings were that casual players prefer easier modes than experienced players regardless of their actual skill, experienced players are more likely to accurately select their optimal difficulty level, and the enjoyment of experienced players correlates with the challenge provided [2].

Adaptive games can also draw from physiological data, such as real-time biometrics. Changchun Liu et al. [53] outfitted players with wearable sensors that measured signals known to correspond to the players' anxiety levels while playing a game provisioned with a DDA system. Using this so-called "affect-based DDA," as compared in the study to a system based solely on gameplay performance, Liu et al. demonstrated that this alternative approach to adaptive game design not only resulted in a more satisfying and challenging game experience but also yielded greater overall player performance.

3.4 Basis for Dynamic Adaptation

Beyond the above examples, there have been many explorations into adaptive games that have targeted different aspects of the player or the player's game experience to find the most useful target metric (or *basis*) for dynamic adaptation. The following is a review of these categories of approaches along with examples:

Performance	Player performance within the game system [5, 15, 39, 60, 68, 71].
Biometrics	Physiological response (e.g., heart rate) [53].
Instruction	Preference or direction given explicitly by the player [13].
Engagement	Attempts to measure engagement or immersion metrics *in situ*, such as through examining eye movement [43] or body language [73].
Personality	Deriving measurements of the player in terms of a psychological or sociological model [19, 36].

Preference	Deriving player predilections through observation of choices made in the game environment [4].
Demographic	Presumptions based on player age, gender, nationality, or other population characteristics [14].
Archetype	Placement of the player in a gamer archetype model, such as those proposed by Bartle or Foster [61].
Behavior	Assessment of player qualities from an analysis of in-game behavior [27, 72].

A basis for dynamic adaptation not heavily represented in current literature is one in which known qualities of the player's capability (in our case, physical capability) are leveraged to alter gameplay. For example, this research proposes that knowledge of the player's height, leg strength, and right elbow flexion could be used as input into an AVG to provide a more optimal experience for an individual player. As another example, consider a puzzle game that employs cognitive metrics such as the player's working memory capacity (i.e., via *n-back* tests) to alter the difficulty or nature of the problems.

This approach is related but distinguished from the other categories listed above. For example, physiological data regarding the player are similar to biometrics; however, by this definition, they refer to relatively immutable aspects of the player that are determined *a priori* rather than at runtime. Similarly, demographic information may in some cases correlate or potentially predict some capabilities, but they are not measurements of the capability itself. Finally, a player's capability profile may indeed affect the player's performance in a particular task within the game, but the metric is regarding that source factor and not a measurement of resulting performance.

In examining the qualities of the target metrics above, we suggest that two key aspects can categorize the metrics. The first is whether the metric can be measured directly ("Empirical") rather than inferred through theory ("Interpretive"). This dichotomy distinguishes dynamic adaptation approaches that can refer to an aspect of the player as ground truth as opposed to those that rely on inference or supplementary theory. The second is whether the metric is correlative and serves as a presumed predictor of the player's quality of interaction with the game ("Antecedent") rather than a measurement of actual engagement consequential to that interaction ("Consequent"). This dichotomy distinguishes the adaptation approaches that rely on assumptive correlations to player flow versus those that attempt to measure flow itself.

The combination of these two spectrums on orthogonal axes can be visualized as depicted in Fig 3.1. With the Empirical-Interpretive range along the horizontal and the Antecedent-Consequent range on the vertical, we can see the groupings under which each of the previously mentioned adaptation bases fall.

For example, a basis for adaptation founded on player biometric data is empirical, because it targets objective physiological data measured by instruments directly monitoring the player. The study mentioned above by Liu et al. [53] evaluated involuntary physiological phenomena such as heartbeat and electrodermal activity to drive a DDA system. This approach is in contrast to a more interpretive personality basis approach such as that employed by Hawkins et al. [36], where players were evaluated on a psychological scale regarding their willingness to take risks based on observations of their decisions. Such a technique is tracking a phenomenon that is dependent on a theoretical, psychological model.

Further, consider a game that adapts based on demographic knowledge of a player. For example, a trivia game might use the age of the player to favor particular decades

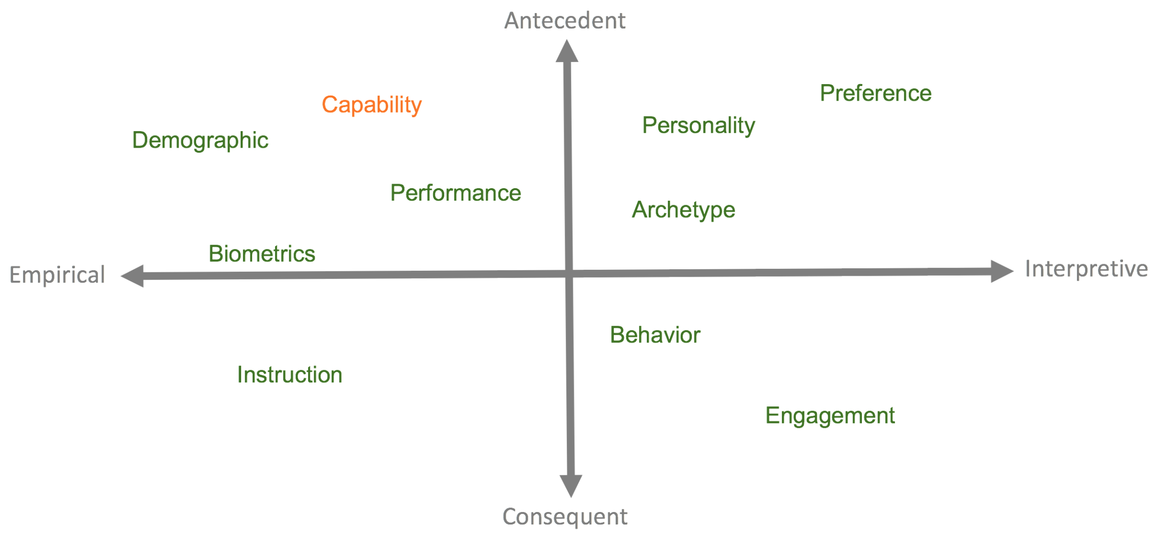


Figure 3.1: A mapping of the bases of dynamic adaptation in games discussed in this section, loosely positioned relative to the degree by which they satisfy the definitions of the two quality axes. Note that in this proposed taxonomy, the categories are not necessarily of equal area, and many may overlap. The approach targeted by SUKI explores the use of known player capability (orange) as an empirical, antecedent basis for adaptation.

when pulling questions regarding music. Like with biometric data, the demographic attributes are nonsubjective, and like both biometric and personality traits, the target metric is one anticipated to correlate to game enjoyment but is not a measurement of that enjoyment itself.

In this way, all of the three aforementioned metric types qualify as antecedent bases according to our definition. While the accessibility of trivia questions or a particular stress response detected in the player may be known to correlate with the player's enjoyment, it is not directly targeting a measurement of flow state or its indicative qualities such as attention, presence, immersion, or self-reported engagement [43]. It is important to note that flow itself is a theoretical phenomenon and cannot be observed directly, but we find distinction in whether the metric directly targets the consequential engagement of gameplay or a proxy metric anticipating that engagement.

For example, consider the approach of Jennett et al. [43], which uses specialized cameras to track pupil movement and dilation in participants engaging in various types of computing activity. They explored potential correlation between gaze trend data (specifically, fixations per second) and a subsequent flow state questionnaire in both immersive and non-immersive conditions to discover significant changes in eye movement patterns as the player becomes immersed in the activity. Though not directly employed toward dynamic gameplay at the time of that study, an adaptive game could use such a metric as an example of an empirical, consequential basis for real-time adaptation.

We derive from the above an opportunity to explore player capability as a basis for dynamic adaptation. The relationship between games and their players is meant to be reciprocal, and this is profoundly realized in the form of adaptive games that can adjust in real time to match the needs of the individual player. Player models are often the mechanism through which adaptive systems are implemented; however, we understand that player traits should not be treated as immutable, but rather qualities that can change over time. This is especially true in the case of physical capability in therapy scenarios where the intent is precisely to modify player capability over time. In the next section, we explore SUKI as an approach to dynamic adaptation based on a mutable player model that can adjust the game's interpretation of player movement to better customize play experience to individual players.

4. PROJECT DESIGN

We understand from the present literature that flow is a key factor in maintaining player engagement and motivation [31], and we also know that player engagement and motivation are the key factors in driving successful patient outcomes in AVG therapy games [65]. Further, in a study applying an experiential model to adaptive games, Kristian Kiili states, “Bad usability decreases the likelihood of experiencing task based flow because the player has to sacrifice attention and other cognitive resources to inappropriate activity” [49]. Therefore, it follows that usability is a factor that may ultimately influence patient outcomes in a therapy AVG.

Therapy AVGs are uniquely sensitive to this issue, where many of the patients play the games to the benefit of precisely the same factors that might disrupt usability. To create engaging and productive therapy games both in service to and in spite of a player’s potential impairment, researchers and developers should seek solutions that maximize accessibility, flexibility, and versatility in body motion input.

Therefore, we believe that beyond providing a solution to the practical problem of improving the flexibility of motion-based input in games, the SUKI system can serve as a useful test bed for examining adaptive game theory. In the previous section, we discussed a multitude of ways in which adaptive games researchers have constructed games that can adapt to player performance, biometric feedback, preferences, personality, behavior, and even demographic information. However, one area not thoroughly covered in existing literature are games that adapt to the known or discovered *capabilities* of the player. This research aims to not only propose a better way for developers and researchers to design input routines for body motion-based video games but also to examine the potential for player capability as a new basis for adaptive games.

4.1 System Overview

The *System for Unified Kinematic Input* (SUKI) system aims to address two fundamental challenges in body motion input programming and therapy AVG development. The first is the inherent complexity of writing abstract routines for human body controls due to the breadth of the domain. The second is the difficulty in designing input processing methods flexible enough to address a broad range of player physiology and therapy goals. In previous sections we examined the current approaches to game-based therapy through commercial exergames (e.g., the Nintendo Wii Fit), therapy-specific AVGs (e.g., Jintronix, VAST Rehab, and Doctor Kinetic), and general input emulation tools (e.g., FAAST). Though many of these offerings provide partial solutions to the fundamental challenges presented, we do not yet have a solution that addresses them all.

Specifically, we desire a system that leverages the full expression of kinematic movement rather than a simple measurement of exercise output, allows for extensible end-user configuration for how player movement drives gameplay without requiring specific knowledge of computer programming, does not limit input configurations to movements predetermined by the developers, enables these changes to be made without the need of source code or recompiling the application, and can adapt its input processing based on the demonstrated individual capability of the player.

SUKI is an input abstraction library that decouples the game's interpretation of input signals from the body motion data reported by the kinematic sensor, where the two are linked by a user-defined text file loaded at runtime. This text file, written in a JavaScript Object Notation (JSON) format, implements a schema that the SUKI system uses to convert measurements of the player's movement to values that can be consumed by the game software at runtime. The games are provisioned at development time with abstract input anchors through which it can read these values

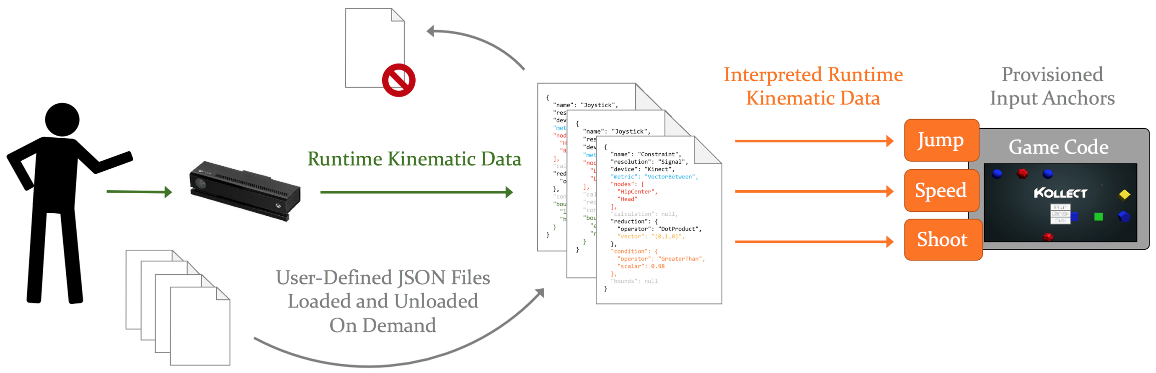


Figure 4.1: Simplified Overview of SUKI. The kinematic data registered by the sensor is not directly interpreted by the game engine as in traditional AVGs. Instead, the game is provisioned with abstract input anchors, and these anchors are served values at runtime according to player movement and the prescribed interpretation defined in JSON schemas created by the end user. Schemas can be customized and loaded or unloaded on demand to change how player movement drives gameplay.

from the SUKI system to drive gameplay. Demonstrated in later sections, the SUKI schema design provides end users with a toolkit for defining custom movements based on angles and distances between areas of the player's body, aiming to be flexible and expressive while not requiring specific programming knowledge or experience. There are no limits to the number of JSON files that can be created by an end user, and they can be loaded and unloaded from the game at any time to change the relationship between player movement and gameplay on demand. An overview of this system is illustrated in Figure 4.1.

4.2 System Requirements

The following discusses the design decisions and implementation of the SUKI system, along with some areas of consideration that inspired and informed its development and SUKI's respective contribution to these areas. It concludes with a list of requirements that guided the creation of the system.

4.2.1 Motion Controller Programming

Traditional game development, namely those that use typical button- or key-based controls, have unique affordances concerning the programming of input routines. In general, the task of writing for abstract input is straightforward, and tools and practices exist for provisioning a game to support an array of controls. Even without developer intervention, operating systems provide system-level tools that enable players to *remap* commands to different keys or even different devices. These solutions provide a substantial amount of flexibility to gaming, particularly for players who may have intervening factors that prevent the use of a particular device or control configuration.

However, similar conveniences do not exist for games that rely on human body movement to convey user intent. Not only are the routines for detecting input fixed to a greater degree, but the domain of expression in analog human movement is larger and vastly more complex than that of a discrete-button controller or even a mouse and keyboard. This issue presents additional complexity when faced with the challenge of adapting a motion-based game for a player, whether attempting to remap the existing controls or to change the therapeutic goal of the activity entirely.

For instance, consider a game in which a player steers a spaceship through an asteroid belt (Figures 4.2 and 4.3). The game was originally written to track the angle of the player's outstretched arms and match the tilt of the arms to the roll of the ship. However, what if a player with limited upper arm mobility wanted to play the game? For example, how could we easily remap the controls to follow the left or right lean of the player's torso, such that it might be valuable as a posture or balance control exercise for a player with Parkinson's disease?

SUKI contributes to this area by providing a solution for achieving this capability enjoyed by traditional games in a new modality. With SUKI, we can gain some of the

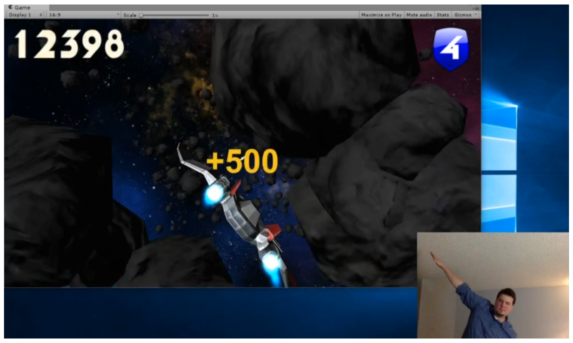


Figure 4.2: Input routines are mapped to specific user input, such as the binding of the ship's roll in *Space Run* to the banking of the player's arms to dodge the asteroids.

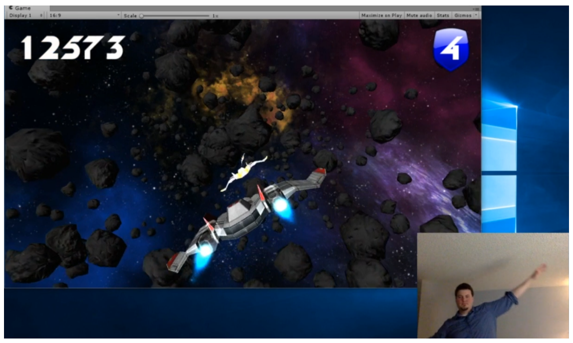


Figure 4.3: As the user banks to the left in *Space Run*, the ship follows, but what if a player with different therapeutic goals wanted to play the game with alternative movement?

same affordances in motion controller games that we currently enjoy in traditional game development.

4.2.2 Game Adaptation

Beyond remapping input controls and re-purposing games for new therapeutic use, another difference between traditional games and motion-based games is the specificity of control. Except for the analog joystick or trigger component on modern controllers, most button presses on a game controller register as a simple, singular event delivered to the program. Keyboard strokes and mouse clicks similarly are either pressed or released. However, movement of the human body is naturally analog, and unlike with a controller, we cannot ensure that every player has the same scope of input potential.

To provide better therapy, developers often provision AVG software with a variety of customizable options and parameters, but even establishing parameters for gesture detection and thresholds for movement assume that a set of pre-programmed input

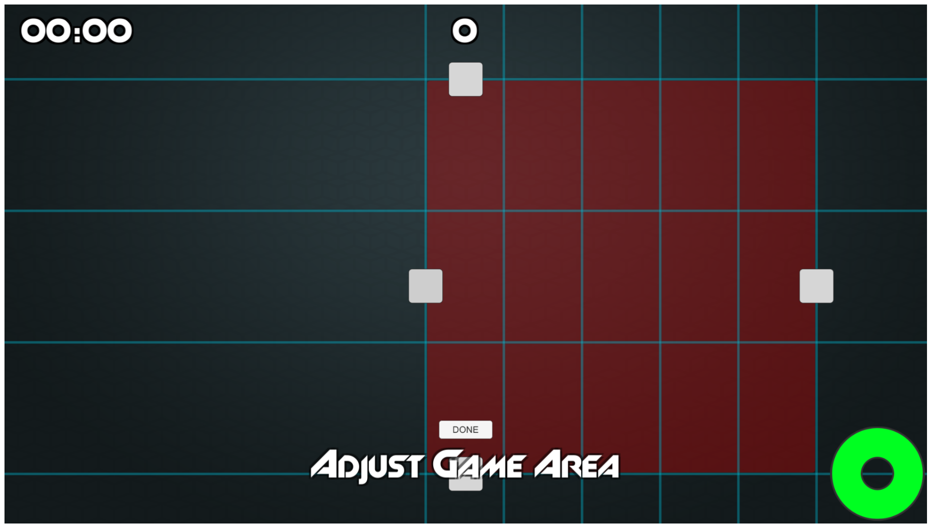


Figure 4.4: An AVG setup screen where a therapist has calibrated the game for a patient with limited use of their left arm. Game activity will take place within the designated area on the right side of the screen to keep gameplay engaging. With SUKI, this calibration zone would not need manual adjustment; instead, it would develop over time after observing the player.

routines exist to watch for those gestures and evaluate those thresholds (Fig 4.4). If a player or therapist ever needed an adjustment the developers had not predicted, they have no means by which they could make the necessary changes. What if the input logic for the game existed outside the application in a way that was configurable without requiring a modification to the game software itself?

Further, since this logic could be changed over time outside the game, what if the software itself could make these changes over time as it began to understand the needs of the player? Though SUKI input configuration files can be managed manually by the therapists and players, our solution will also demonstrate the use of adaptive game techniques to fine-tune the challenge of the activity based on a knowledge of the player's capabilities as recorded in their unique profile.

SUKI contributes to this area by providing a method for adapting a game's input requirements based on knowledge gained regarding the player both ahead of time and during gameplay.

4.2.3 SUKI Requirements

When considering the requirements for a satisfactory solution to the AVG development challenges discussed, our design aims to implement six key features:

1. **Abstract Input**: Enable developers to write input routines that are abstracted and decoupled from the particular motion capture device or anticipated gesture.
2. **No Recompiling**: Enable therapists to modify the gameplay experience toward customized therapeutic goals without any change to the game's code.
3. **Patient Individualization**: Allow the input to be tuned or calibrated to meet the physical capabilities of the individual patient.
4. **Compound Metrics**: A modular, flexible system that allows for the application of multiple schemas at once.
5. **Runtime Updates**: Schemas can be updated at runtime to affect gameplay immediately.
6. **Ease of Integration**: A portable, reusable software library that new and existing AVG projects can easily import and integrate.

4.3 System Architecture

The following is an overview of the architecture of the SUKI framework from the perspective of the developers who may implement it in their applications and the users who may configure custom schemas to address their individual therapy goals.

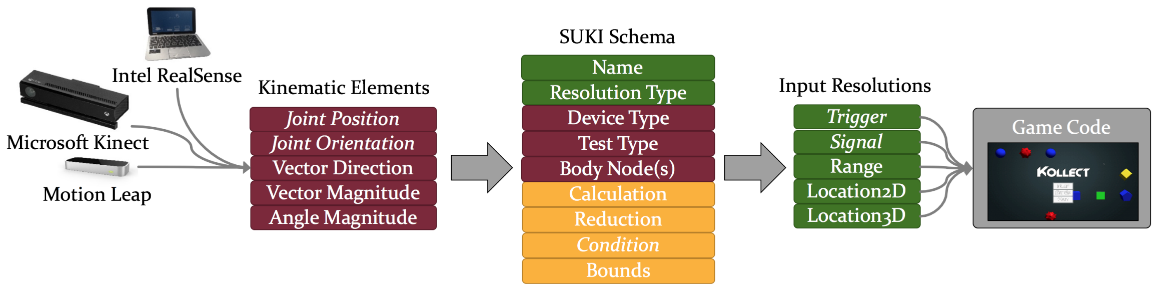


Figure 4.5: Simplified Overview of SUKI System Architecture. Input Resolutions (green) in the rightmost column are defined by developers and integrated into the game. Kinematic elements (burgundy) in the left-most column filter body data from multiple sensors to a unified format for system processing. The SUKI Schema in the center column binds the two together and adds Operators (gold) that can manipulate the data as it passes between the body motion data sensor and the game software.

4.3.1 Overview

The SUKI system consists of three primary components as illustrated in Figure 4.5. The first is a definition of *Input Resolutions*, a set of abstracted user signals with which developers can provision their game in anticipation of integration with motion-based inputs. The second is a codified collection of *Kinematic Elements*, an organizational set of the information a program can expect to receive from a kinematic device. The third is a set of flexible *Operators*, which describe the manipulations that should be performed on the Kinematic Elements before they resolve into input commands. The particular relationships among all components are defined by a *SUKI Schema* (currently implemented as a JSON string), which can be loaded and unloaded on demand to alter the game's behavior concerning player input.

The following sections will discuss Input Resolutions, Kinematic Elements, and SUKI Schema components (including Operators).

4.3.2 Input Resolutions

To determine how the modified signals from the kinematic sensors will resolve to commands within the game, we must consider the fundamental categories of information that users convey to computer systems to implement control and indicate their intent to that system. To address this question, we seek to organize the types of information that a typical game would require from a player, regardless of the device or modality that facilitates that communication.

After reviewing input requirements from several AVGs and games in general, we derived the following categories of input through which players register their directions to the software. All user intents (or *Input Resolutions*) supported by SUKI are presented in the following list:

Inputs Conveying Timing & State Information:

Trigger A simple event that "fires" once and is immediately reset. A trigger does not carry additional data but by its nature inherently conveys temporal data concerning the moment in which it fired. An example of a trigger input used in games is a command from the player for their avatar to jump at a specific moment in time, perhaps by pressing a button at that same moment.

Signal A trigger that is not necessarily reset after it fires, but must be explicitly reset. A signal can be used to indicate a binary state (on or off) over time and can also be regarded as a *compound trigger* (i.e., two signals that indicate a change in state). An example of a signal may be a command from the player for their avatar to sprint, perhaps instructed to begin with a button press and end when the button is released.

Inputs Conveying Numerical Data & Value Information:

Range A value on a continuum between two bounds. SUKI normalizes range values to a floating point value between zero and one (inclusive). An example of a range may be the degree to which the player commands a virtual vehicle to turn, perhaps as conveyed by the horizontal offset of an analog joystick.

Location A coordinate for a position in 2D or 3D space. A location can also be regarded as a *compound range* (i.e., multiple range values in aggregate). Like ranges, in SUKI all location axis values are normalized to floating point values between zero and one (inclusive). An example of a location input may be a command from the player to fire their weapon at an object on the screen, perhaps communicated as a mouse click or a touch on the screen at that object's location.

This set of input resolutions attempts to be comprehensive to serve the player control needs in a video game; however, additional categories may exist. For example, one could argue that array information (e.g., a string representation of a voice command or an audio stream produced by singing) could be used as input into a game but could not be implemented using the categories above. Conversely, some of these proposed counterexamples could be regarded as a compound or aggregated grouping of the types listed above. Further examination and addition of one or more of these fundamental types may enhance the coverage and applicability of this taxonomy.

By invoking these four input types, developers can declare input resolutions in their software with unique names that register in the system and become available by named reference. With a SUKI library built for the Unity game development engine, developers will typically query for the values of their provisioned inputs in an Update() or FixedUpdate() function called each time the graphics engine or physics engine renders a new frame of the game. Listing 4.1 shows an example of what the developer code might look like when querying the SUKI library for the values of provisioned input anchors for each of the above types.

```
void Update() {
  // cause the character to jump if triggered
  if (Suki.GetTrigger("Jump")) {
    mainCharacter.Jump();
  }
  // raise/lower shield depending on signal state
  bool shouldRaise = Suki.GetSignal("ShieldUp");
  if (shouldRaise && !shield.IsRaised) {
    shield.Raise();
  } else if (!shouldRaise && shield.IsRaised) {
    shield.Lower();
  }
  // shoot fireball at speed queried from range input
  if (Suki.GetTrigger("Shoot")) {
    float firePower = Suki.GetRange("Firepower");
    StartCoroutine(ShootFireball(firePower));
  }
  // move a pointer on the screen
  Vector2 2Dtarget = Suki.GetLocation2D("Pointer");
  pointer.transform.position = ScreenCoordinates(2Dtarget);
  // move light source to the 3D location
  Vector3 3Dtarget = Suki.GetLocation3D("Light");
  light.transform.position = WorldCoordinates(3Dtarget);
}
```

Listing 4.1: Sample Code Referencing SUKI Input Resolutions

4.3.3 Kinematic Elements

Within the SUKI library, we use the term *node* to describe one of many pre-defined locations on the body tracked by a motion input sensor. For example, the Microsoft

Kinect v2 tracks 25 individual nodes (called "joints" in the Kinect nomenclature) [54]. The player *skeleton* refers to the full set of all nodes reported by the sensor.

We use the term *frame* to describe a snapshot of the player's kinematic data at a particular moment in time. The frame contains information for the entire skeleton, including positions and orientations of every defined node. Motion input devices usually capture these frames at a constant rate. For example, the Microsoft Kinect v2 processes frames at 30 Hz [52].

The following are the fundamental *Kinematic Elements* of the player skeleton on which SUKI can perform measurements each frame:

Node Position	A vector describing the position (x,y,z) of a node in 3D space. For example, this might track the 3D coordinate location of the player's head.
Node Orientation	A Euler angle describing the orientation of a node in 3D space. For example, this might track the direction the player's right palm is facing.
Vector Between	A vector describing the relative position (x,y,z) of one node to another. For example, this might track the vector formed between a player's hand positions in 3D space. From this value, we can measure either the direction between the nodes or examine the scalar value (length) to determine the distance between the two nodes.
Angle Between	The angle formed between two nodes relative to a third node, thereby forming a *node triad*. For example, a node triad may be formed by the right shoulder and right wrist nodes with a connection at the right elbow node. This measurement would cal-

culate the angle formed between the RightElbow ➔ RightShoulder vector and the RightElbow ➔ RightWrist vector, thereby yielding the flexion of the player's elbow. Though most often constructed using nodes adjacent to each other on the skeleton, SUKI supports node triads of any three arbitrary body nodes (e.g., left hand and right hand positions relative the head) if noted explicitly in the schema.

4.3.4 SUKI Schema Components

The Input Resolutions and the Kinematic Elements are brought together at runtime by a *SUKI Schema*, which exists as a simple JSON string retrieved from a database or read from a file located on disk alongside the compiled game binary. The user specifies the file (or files, in the case that they deploy multiple schemas simultaneously) at runtime. Schema files can be edited, removed, and reloaded on demand, enabling the player or therapist to customize as needed the way the engine interprets input to the individual player's physical capabilities and therapeutic goals.

As the schemas attempt to distill raw player skeleton data into signals of the player's intent, each one defines a discrete player movement that will potentially resolve into an input signal for the game. SUKI implements these movement definitions as a collection of several attributes:

A) The number of nodes it observes (single or multiple)

B) The aspect or relationship of the node(s) it evaluates (position, orientation, vector-between, etc.)

C) The type of *Input Resolution* it will target (range, etc.)

D) The metric of the aspect or relationship it will evaluate (magnitude, direction, x-component, etc.)

E) The observed metric's *data type* (i.e., vector, scalar)

Additionally, there are a few special considerations that we may have for particular schemas, depending on their construction:

F) A preparatory adjustment required for the metric prior to evaluation (offsets, scaling, etc.)

G) An action to be taken on the resulting value (signal setting, value normalization)

These attributes are further organized and expanded upon below to explain the design of the SUKI schema structure.

A & B - Node Metric

The number of nodes we observe and the features of those nodes we are evaluating are inextricably tied; therefore, we combine the A and B aspects listed above into a compound metric called the *Node Metric*. The Node Metric, or the combination of node count and characteristic of the node(s) measured, classifies every SUKI schema into one of four categories:

Position - Vector describing position of a single node

Orientation - Vector describing orientation of a single node

VectorBetween - Vector describing relationship between two nodes

AngleBetween - Scalar value describing the angle in a node triad

C - Input Resolution

The second defining element of a schema is its *Input Resolution*, which determines which fundamental input representation the schema will render in the application. Though discussed earlier in Section 4.3.2, they are listed here again to emphasize the type of data that they generate:

Trigger - An event with no additional data other than the time it was created

Signal - A boolean value indicating whether a state is active or inactive

Range - A floating point value between zero and one

Location2D - A 2-dimensional vector

Location3D - A 3-dimensional vector

A & B & C - Node Metric + Input Resolution

We further classify each schema by a joining of its Node Metric and its Input Resolution. For example, a schema that raised a signal when the player's right hand was within 0.5 meters of his or her head would be a "VectorBetween Signal" schema. A schema that fired an event whenever the player's left palm faced upward would be an "Orientation Trigger" schema. This categorization results in a total of 18 valid schema subtypes, with two of the potential permutations excluded because the AngleBetween node metric yields a scalar and is not eligible for location resolutions. Table 4.1 displays the Node Metric and Input Resolutions that may be combined to form a valid schema.

Table 4.1: Node Metric + Input Resolution Schema Compatibility

	Position	Orientation	Vector Between	Angle Between
Trigger	✔	✔	✔	✔
Signal	✔	✔	✔	✔
Range	✔	✔	✔	✔
Location2D	✔	✔	✔	✗
Location3D	✔	✔	✔	✗

✔ - Schema is possible
✗ - Schema is not possible

D & E - Reduction

For every Node Metric + Input Resolution permutation, we must declare what element or aspect of the node metric we want to observe and resolve to an input intent. This decision will also require us to determine the data type of the resolution; for example, a trigger or signal will expect a boolean (to fire in the former or to indicate the current state in the latter), a range will require a floating point value, and so on.

The schema's *Reduction* operator is responsible for performing this conversion from a vector-based schema to a scalar-based resolution. For example, consider a game avatar with variable running speed that the therapist wishes would be faster the further the player extends a leg to the side outward from his or her body. The appropriate schema might monitor the vector between the player's pelvis and right foot nodes but would want to reduce the vector to just its 'x' component before yielding a range input value. Similarly, consider the laser gun on a spaceship that the therapist wishes would only fire when the player rotates his or her left palm toward the screen (and sensor). The schema might monitor the orientation of the player's left hand but would want to calculate the dot product between the hand's orientation and the forward vector $\hat{k}$ to yield a scalar value indicating how closely they are aligned.

Note that schemas with location data resolutions cannot employ a reduction operator because we do not want to reduce our node metric's output vector to a scalar value. Also, because the node metric for an AngleBetween schema is already a scalar, it also cannot use a reduction operator.

F - Calculation

Before the reduction is applied, we can perform transformations on the node metric's vector using a *Calculation* operator, which could normalize or re-center a vector, find a cross product, or scale the vector before other operators are meant to process it. Calculations apply only to vector node metrics, so they are not available on AngleBetween schemas. However, they are available (and optional) for all input resolutions on all other schemas.

An example of a pre-reduction vector calculation would be subtracting a z-value of four to re-center a position vector to four meters in front of the sensor. Another example might be to calculate the cross product of a vector between two nodes and the upward vector $\hat{\jmath}$ to isolate a patient's limb movement within a physiological plane.

G - Conditionals & Bounds

Triggers and signals require a test (or *Conditional* operator) to determine whether or not they should respectively fire or raise. Similarly, ranges must be normalized to a value between zero and one using the extents defined in a *Bounds* operator. The same applies to location (both 2D and 3D) resolutions so that their components lie between zero and one.

An example of a conditional might be to fire a trigger if a distance between two nodes drops below a threshold. A schema's bounds field might call for normalization of the angle of the elbow (e.g., bounded at 30-150 degrees) to a range value between zero and one.

4.3.5 SUKI Schema Restrictions

The previous section discussed the information required by a schema to effectively associate the input resolutions provisioned in the game by the developer with the kinematic elements available via the body motion input device while providing the flexible elements (operators) required to define adjustable relationships between them.

However, as noted, not every schema type can employ each of the four operators. Table 4.2 notes which components are required, restricted, or optional with each Node Metric + Input Resolution permutation.

Table 4.2: Node Metric + Input Resolution Operator Compatibility

	Position				Orientation				Vector Between				Angle Between			
	%	▼	?	||	%	▼	?	||	%	▼	?	||	%	▼	?	||
Trigger	✱	✓	✓	✗	✱	✓	✓	✗	✱	✓	✓	✗	✗	✗	✓	✗
Signal	✱	✓	✓	✗	✱	✓	✓	✗	✱	✓	✓	✗	✗	✗	✓	✗
Range	✱	✓	✗	✓	✱	✓	✗	✓	✱	✓	✗	✓	✗	✗	✗	✓
Location2D	✱	✗	✗	✓	✱	✗	✗	✓	✱	✗	✗	✓	✗	✗	✗	✓
Location3D	✱	✗	✗	✓	✱	✗	✗	✓	✱	✗	✗	✓	✗	✗	✗	✓

Component Type:

% Calculation - Converts a vector into another vector
▼ Reduction - Reduces a vector to a scalar
? Conditional - Evaluates a scalar to produce a boolean
|| Bounds - Normalizes a vector or scalar to the same type

Component Type:

✓ - Schema must have this operator
✗ - Schema cannot have this operator
✱ - Schema can (but does not have to) have this operator

4.3.6 SUKI Schema Definition

Listing 4.2 contains the basic JSON definition for the SUKI Schema. As discussed later, some operator fields have additional sub-fields that engage the profile and adaptation engine.

```
{
  "name": "UniqueIdentifier",
  "resolution": "ResolutionType",
  "device": "DeviceName",
  "metric": "MetricType",
  "nodes": ["NodeName", ...],
  "calculation": {
    "operator": "VectorOperator",
    "vector": {"x":0.0, "y":0.0, "z":0.0}
  },
  "reduction": {
    "operator": "ReductionOperator",
    "vector": {"x":0.0, "y":0.0, "z":0.0}
  },
  "condition": {
    "operator": "ScalarOperator",
    "scalar": 0.0,
    "percentage": 0.8
  },
  "bounds": {
    "low": 0.0,
    "high": 0.0,
    "extents": Boolean
  }
}
```

Listing 4.2: SUKI Schema JSON Definition

Implemented as a simple text file, the elements of the SUKI schema are available for modification by the end-user (player or therapist) and enable customization of the

Table 4.3: SUKI Schema JSON Fields

Name	A string that will uniquely identify this schema at runtime.
Resolution	The type of input resolution to which this named schema is tied.
	“Trigger” - A single timed event.
	“Signal” - A boolean indicating whether a state is active.
	“Range” - A floating point value between 0 and 1.
	“Location2D” - a 2D vector (x,y).
	“Location3D” - A 3D vector (x,y,z).
Device	The type of kinematic sensor.
	“Kinect” - The Microsoft Kinect v2 sensor.
	“Leap” - The Leap Motion Controller sensor.
Metric	The metric from the kinematic device node(s) to evaluate.
	“Position” - The location of a node in 3D space (vector).
	“Orientation” - The orientation of a node (vector).
	“VectorBetween” - The vector between two nodes (vector).
	“AngleBetween” - The angle formed in a node triad (scalar).
Nodes	One or more body nodes defined by the device (e.g., “Head”).
Calculation	Performs a vector calculation on the initial metric.
	“CrossProduct” - cross product against an operand vector.
	“Add” - add to an operand vector.
	“Multiply” - scale by an operand vector.
Reduction	Reduces a vector value to a scalar.
	“DotProduct” - dot product against an operand vector.
	“X/Y/ZValue” - extract the vector component.
Condition	If resolution is a trigger/signal, defines the value threshold.
	“GreaterThan/Equal/LessThan” - compare value to a scalar.
Bounds	If resolution is a range/location, defines the normalization.

gameplay experience without any change to the game software. Table 4.3 contains a description of the elements and applicable values recognized by the system for each.

The developers set the name and resolution fields that associate the schema with an input anchor they have provisioned in the software. The device, metric, and nodes fields tie the schema to the kinematic elements established in the SUKI library integrated by the developer within the game. What remains are the calculation, reduction, condition, and bounds operators, which provide the additional flexibility and expressiveness required to satisfy the large number of potential end-user scenarios.

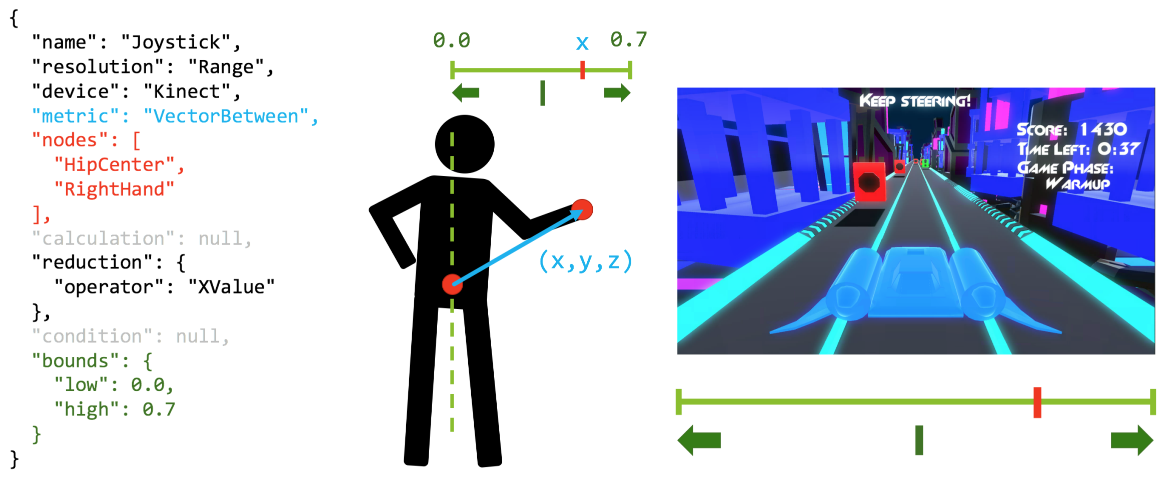
4.4 Player Modeling and Adaptation

A flexible input system only partially solves the problem of individualizing gameplay experiences for every user according to their therapeutic goals. Among players that may be seeking the same type of exercise, and therefore the same physical movements tied to gameplay, there remains variance in their personal abilities to negotiate and execute those movements that must be considered. This consideration constitutes the other half of the problem we address for motion-based input control systems, which not only struggle with providing flexibility in linking player motion to gameplay but are affected much more by variance in player physiology than traditional computer input devices like a keyboard or game controller.

Therefore, in pursuit of maximizing flow and player motivation in AVGs, we investigate a player modeling strategy that engages the potential for SUKI's adaptability beyond its schema flexibility and the types of movements it can interpret. Turning our focus toward adaptive game systems and player capability as a basis for dynamic adaptation, we explore how SUKI might incorporate knowledge of the individual player in its measurements and analysis to provide a more individualized and targeted experience.

4.4.1 Overview

Consider a schema that measures a player's lateral reach and converts it into a range value called "joystick" in a spaceship flying game to steer the ship, as illustrated in Figure 4.6. To perform the range conversion, the schema requires a defined bounds within which to normalize the distance the player was able to reach. Working with a 13-year-old adolescent girl with reduced upper body motility, the therapist sets this bounds extent to 0.7 meters as an optimal and challenging range for that patient. The patient plays the game, pulling her arm inward and pushing it out to the side



```
{
  "name": "Joystick",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [
    "HipCenter",
    "RightHand"
  ],
  "calculation": null,
  "reduction": {
    "operator": "XValue"
  },
  "condition": null,
  "bounds": {
    "low": 0.0,
    "high": 0.7
  }
}
```

Figure 4.6: Example schema converting the player's lateral reach into a range input that controls the ship's lateral movement.

while leaning to control the ship on screen.

Over time, the therapist may encounter two issues. The first is observed when the therapy improves the patient's balance and motility over time, enabling her to reach farther than the original extent set in the schema. The second is encountered when the therapist deploys the same schema for new patients with similar conditions and therapeutic goals. One of the new patients is an exceptionally tall adult male and is not challenged by the same distance setting, and the second is an adult female who experienced trauma in her elbow and finds that normalizing the play range over 0.7 meters is far too challenging. In all cases, the therapist could manually update the schema file, but an adaptive system based on knowledge of the player not only would improve the schema's usefulness across patients but would ensure that it provided an optimal and challenging experience *within-patient* even as that individual patient's capabilities changed over time.

The adaptation feature provided by SUKI addresses this problem. As illustrated in Figure 4.7, this system consists of two components. The first is a persistent player

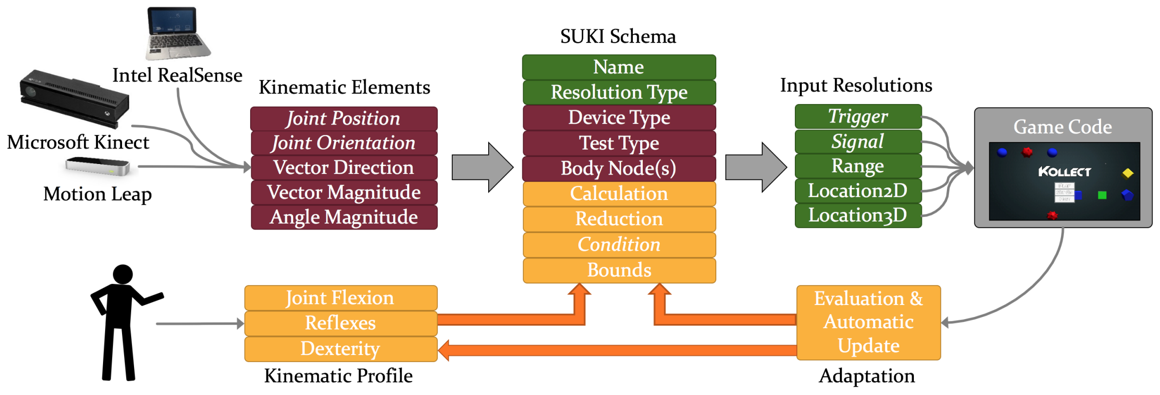


Figure 4.7: Extended SUKI System Architecture. The operators within the SUKI schema provide flexibility in the input interpretation and a method for implementing game adaptability. The kinematic profile tracks player capabilities persistently and uses this data to set bounds and condition values in the schema appropriate for that player. The adaptation system detects changes in player capability over the course of play sessions and updates both the schema and the persistent player profile.

model that reflects the player's capacity to perform a schema's movement, and the second is a monitoring system within the game that can detect changes in this capacity and use this information to update the schema's interpretation and the player's model. These components can influence the system via the operators in the SUKI schema and are both explained in further detail in the sections below.

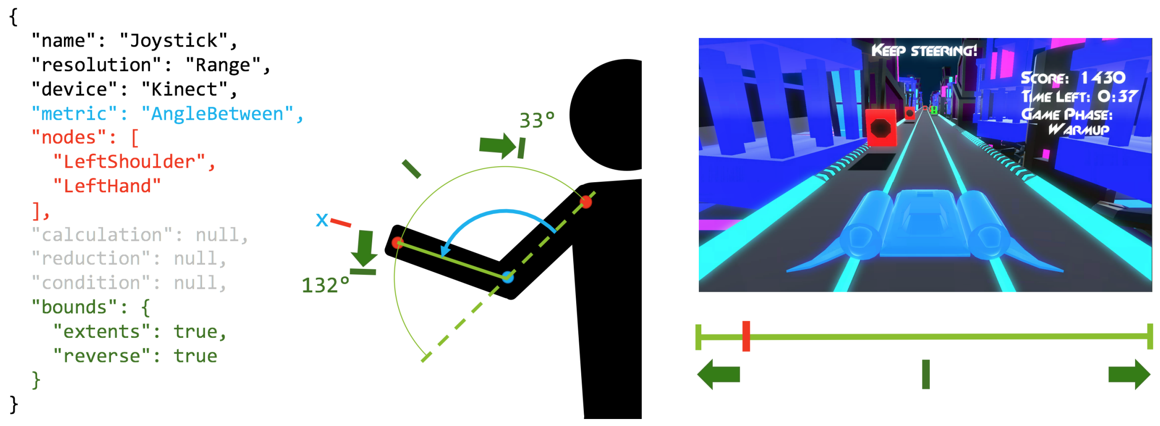
4.4.2 Kinematic Profiles

The SUKI kinematic profile differs from typical player models in that it does not consider player preferences or predilections [6, 77, 76], nor does it examine a player's motivation [29]. However, keeping in line with Heeter's broader definition for player models [37] we examine player traits, specifically those related to the player's anticipated capacity to perform the movement defined by a schema. Fully realized, we might someday take direct measurements of the player's joint flexion, reflexes, fine motor accuracy, and other determinants of capability that we can use to derive an estimate for player capacity in the schema's defined movement. However, in its current implementation, we track the player's *extents* relating to the minimum and maximum values demonstrated or achieved for a schema-specific movement.

Therefore, capability profiles are currently unique for each player and schema combination; they are not at the moment a generalizable piece of information regarding the player, but by their nature are specific to the domain of the schema. Consider a scenario following from the side-reach example above in which a therapist works with multiple patients partaking in elbow flexion rehabilitation. The therapist creates the schema illustrated in Figure 4.8, where instead of measuring lateral reach, patients will change the angle of their left elbows (in this case independent of arm orientation), where joint extension will push the ship left, and flexion will pull the ship right.

Understanding both that different patients will have varying elbow flexion and that this capability will ideally improve in each patient over time, the therapist chooses to reference kinematic profile values instead of literal values to define the bounds within the schema. From observations made by SUKI during during play, the profile for an adolescent female trauma rehabilitation patient knows that she can currently extend her elbow to 132 degrees and can flex to 33 degrees. These values become the target range for the angle normalization, where the patient can play the game and engage in therapy until her capable range increases. A second patient demonstrates a maximum extension of only 104 degrees, but the unique kinematic profile associated with that patient will provide SUKI with an appropriately reduced normalization range.

In the current implementation, player profiles are stored in a database and retrieved via authenticated endpoints securely and remotely at the start of each play session. Persistence of the profile data in a remote server enables the profiles to be accessed and applied to any session the patient plays, regardless of location (e.g., at the clinic or in the home). This implementation is additionally valuable when we



```
{
  "name": "Joystick",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "AngleBetween",
  "nodes": [
    "LeftShoulder",
    "LeftHand"
  ],
  "calculation": null,
  "reduction": null,
  "condition": null,
  "bounds": {
    "extents": true,
    "reverse": true
  }
}
```

Figure 4.8: Sample schema measuring the angle of the left elbow as a range to steer a spaceship. Instead of defining normalization bounds as literals, the schema references *extents*, which will substitute the player's personal values in the runtime calculation. Note the use of the "reverse" flag in this schema used on the left side, where a large angle value will resolve to a small range input to the game (i.e., move the ship left), and vice versa.

consider how existing profiles for one schema might inform expectations for additional schemas and even apply across games when engaging in a new activity.

Though this is a topic intended for future work, the idea of existing extents informing future schema values is partially realized in the fact that schemas define a distinct player motion that is applicable anywhere in a game where there exists an input resolution of the same type. For example, the schema for normalized left elbow flexion targets a range input called "joystick" in Figure 4.8. However, in another game where a range input is meant to drive the force of a golf swing or the speed of a running avatar, the same schema could be applied by simply updating the name field to that provisioned by the developers for that input anchor. In this way, schemas can apply across games to link the same player motion to new gameplay mechanics, and the individual player profile will similarly transfer to the new game as well.

4.4.3 Adaptation System

The SUKI adaptation system differs from traditional DDA systems in that it does not examine player's performance relative to game objectives as a basis for adaptation. Additionally, it does not examine interpretive metrics regarding flow state or measure theoretical traits of the player such as personality-based approaches. Instead, SUKI uses empirical, *antecedent* metrics regarding player capability as a proxy measurement for challenge and engagement, anticipating that aligning player ability to the challenge of the game will increase the likelihood for an optimal player experience [20].

Though therapy AVGs aim to leverage the intrinsically motivating element of games toward therapy and consequently have an incentive to maximize player motivation and engagement within the game, the underlying goal of a therapy AVG is ultimately to provide effective therapy. Therefore, we believe an adaptation system that uses player capability as its basis is, in this case, more appropriate than one that merely observes in-game performance or behavior that is both limited to the domain of the game and a *consequence* of the interaction (rather than a predictor for that interaction). This decision also shares a foundation in therapy science, where we understand that therapeutic goals are optimally pursued by continuously tailoring therapy to the changing condition of a patient, especially in the case of chronic disabilities [55].

To that end, while player profiles provide flexibility in managing schemas and applying them to patient sessions, they would be limited if not provided the means to adapt to changing player capability. Therefore, we have included in the SUKI framework a component that monitors the player's demonstrated movement capability within the context of a schema to update the game and the persistent kinematic profile in real time as new physical movement extents are demonstrated to have been achieved by the player.

Consider the examples illustrated in Figures 4.9 and 4.10, where a therapist working with an elderly male patient with Parkinson's disease wishes to focus on upper body movement and posture simultaneously. In a game where the developer has provided a trigger input to fire the ship's main cannon, as a best practice they have also provided an optional signal input as a *constraint* that locks the cannon if a kinematic condition is not met. The therapist deploys two schemas concurrently: a trigger based on the patient raising his right arm above his head and a signal based on the player standing up straight without hunching forward or leaning to the side.

In the first schema, a trigger input fires when the player reaches his right hand above his head beyond a threshold determined by his kinematic profile (80% of demonstrated maximum). The second schema, used as a constraint, activates a signal input only when the patient is standing up straight. The use of the dot product reduction against the upward vector $\hat{j}$ allows this constraint to function whether the player is leaning forward, back, or to either side. The software is programmed to trigger a notification and gameplay penalties unique to the game when the constraint is not satisfied (i.e., the constraint signal is not active); in this case, the ship's cannon will not fire regardless of the right hand's position. With explanation from their therapist regarding the nature of the schema, this compels the player to maintain an upright posture throughout the session while engaging in other directed movements.

As in the previous section, the first schema makes use of the extent reference when setting the threshold condition for the trigger input. By default, this would establish the trigger to only fire when the player exceeds their previously demonstrated maximum value for vertical reach. However, to promote more achievable gameplay goals, the therapist has engaged an optional field in the schema available to the condition operator that sets the threshold to 80% of the maximum extent recorded in the player's kinematic profile.

```
{
  "name": "Fire",
  "resolution": "Trigger",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [
    "Head",
    "RightHand"
  ],
  "calculation": null,
  "reduction": {
    "operator": "YValue"
  },
  "condition": {
    "operator": "GreaterThan",
    "percentage": 0.8
  },
  "bounds": null
}
```

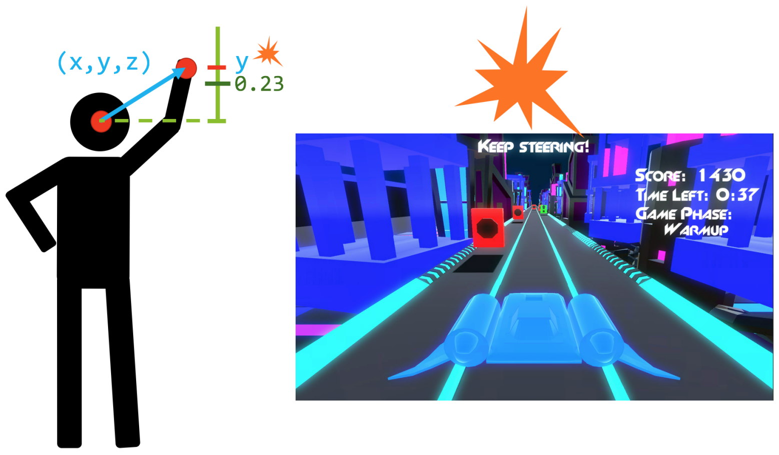


Figure 4.9: Two schemas are deployed concurrently in a session with a patient with Parkinson's disease. The first schema fires a trigger input within the game whenever the player raises their right arm above their head over a variable threshold (80%) of the player's previously demonstrated capability.

```
{
  "name": "Constraint",
  "resolution": "Signal",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [
    "HipCenter",
    "Head"
  ],
  "calculation": null,
  "reduction": {
    "operator": "DotProduct",
    "vector": "(0,1,0)",
  },
  "condition": {
    "operator": "GreaterThan",
    "scalar": 0.98
  },
  "bounds": null
}
```

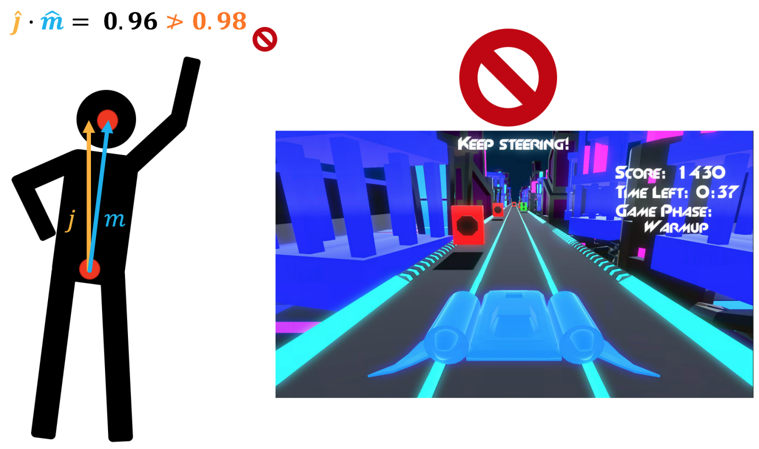


Figure 4.10: An additional schema establishes a constraint to ensure that the player maintains correct posture while performing other motions. If the player fails to maintain proper posture, the game will alert the player and engage game-specific restrictions or penalties to encourage the player to correct the constraint.

As the player improves, new extents will be reached, thereby increasing the height at which this condition will fire the trigger. The SUKI adaptation system evaluates the resulting value from every schema calculation throughout the play session, and if the player achieves a new extent, that value will be updated in the current session as well as sent to the database for recording in the player's persistent kinematic profile. When the player starts a new session in the future, this updated value will serve as the starting benchmark for evaluating the trigger condition.

Some conditions for which individuals receive therapy are progressive, and it is reasonable that over time capability may not improve but instead decline. Over the long term, this may apply to certain traits even in players without any particular disability. In the current implementation, the kinematic profile for each player begins without assumption and determines its values through observation of the player; therefore, a reset of the kinematic profile for a player experiencing a reduction in capability serves as an effective solution at present. Currently, therapists can perform this operation through an administration tool that interfaces with the database. In the future, the adaptation system may even be capable of detecting these regression trends over time and modify player expectations automatically.

Together, the persistent kinematic profile and adaptation system enable flexible schemas that support multiple users without the need to change the schema. They not only facilitate the tailoring of gameplay experience to every individual player, but they will continue to update that experience to match the player's capability as they achieve new therapeutic goals. Finally, with the ability to reuse schemas across games, this profile will be activated whenever the player engages this exercise movement, regardless of the particular game they are playing.

5. EVALUATION

We present the effectiveness of SUKI in the discussion of three case studies, which serve as an evaluation of the solution to fulfill the contribution aims and address the problems proposed at the beginning of this document. The first case study is an example of using SUKI to construct a shoulder movement therapy AVG for patients with cerebral palsy (CP). The second is one in which that same game is re-purposed via the SUKI system toward an alternate therapy for an entirely different patient group. The third is one in which an existing therapy AVG developed traditionally with hard-coded input routines is retrofitted with SUKI and schema-based, adaptive input functionality to explore SUKI's potential for enhancing existing games.

In examining the case studies and the examples previously illustrated, we evaluate our solution based on the requirements outlined in Section 4.2 and explicitly enumerated in Section 4.2.3 that we derived from our investigation of the problem space. Specifically, the first two case studies will demonstrate SUKI's capacity for providing a system that enables (1) Abstract Input with (2) No Recompiling that allows (3) Patient Individualization with support for (4) Compound Metrics that can be (5) Updated at Runtime. While these two cases will also demonstrate the (6) Ease of Integration into a game at design time, the third case study will further demonstrate the ease of integrating SUKI into an application that has already completed its development.

5.1 Case Study 1: Constructing a Therapy AVG with SUKI

As demonstrated in a few of the design examples, SUKI was selected for implementation in the AVG therapy game *Citadel*, a futuristic "endless runner" in which players

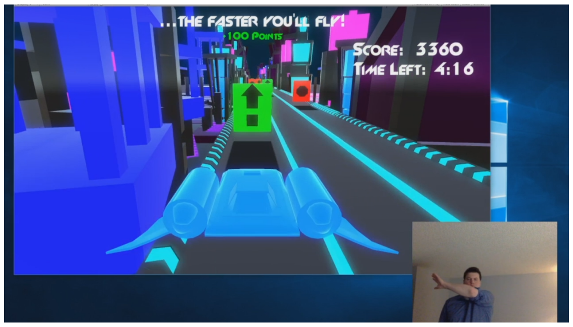


Figure 5.1: In *Citadel*, the player guides a hovering spaceship along a three-lane track. In this scenario, the ship is "locked" to one of the three lanes, where the player must move discretely among them.

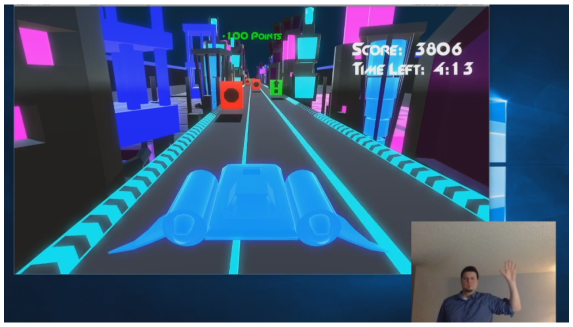


Figure 5.2: Players move to hit the green blocks for points and dodge the red blocks that halt the ship's progress. In this scenario, the ship is controlled in an analog fashion and is not locked to a lane.

guide a spaceship forward down a three-lane track, dodging obstacles and picking up power-ups (Figures 5.1 and 5.2). Designed for patients with CP, the Kinect-based exercise game encourages the use of the upper body to promote strength and movement planning.

5.1.1 Developer Integration

The primary control objective in *Citadel* is the movement of the ship left and right while it is automatically propelled forward down the track. To allow for maximum flexibility, we provisioned three movement modes within the game. The first mode treats each lane as a discrete position, where the player can input a "Placement" range value to indicate which of the three lanes they want the ship to fly in (i.e., 0.0-0.33, 0.33-0.67, 0.67-1.0). The second also views the track as three discrete lanes, where it moves between any two of them when receiving a "MoveLeft" or "MoveRight" signal input. The third considers the full track as an open field where the player can move in a more analog fashion anywhere they please (though the green and red blocks will still be generated to align with one of the three lanes). In this last case, the game reads a range value called “Joystick” and will move the player left and right according to the value’s distance from the center (i.e., <0.4 to move left, >0.6 to move right, with a 20% “dead zone” in the center). To implement these movement modes, the developers include the code from Listing 5.1 in their main game loop.

```
if (suki.RangeExists("Placement")) {
  // range value to define three placement lanes
  float range = suki.GetRange("Placement");
  float leftMoveX = .33f;
  float rightMoveX = .66f;
  // move to L/M/R depending on the range value
  if (range < leftMoveX) {
    MoveToLane(Position.Left);
  } else if (range > rightMoveX) {
    MoveToLane(Position.Right);
  } else {
    MoveToLane(Position.Middle);
  }
} else if (suki.SignalExists("MoveLeft") &&
           suki.SignalExists("MoveRight")) {
  // pair of trigger values to move left or right
  bool moveLeft = suki.GetSignal("MoveLeft");
  bool moveRight = suki.GetSignal("MoveRight");
  // add a cooldown to the signals
  timeSinceLastLaneMove += Time.deltaTime;
  // if there is a distinct direction signal active
  if ((!moveLeft && !moveRight) || (moveLeft && moveRight) ||
      (timeSinceLastLaneMove < LaneCooldown)) {
    return;
  } else if (moveLeft) {
    if (position == Position.Middle) {
      MoveToLane(Position.Left);
    } else if (position == Position.Right) {
      MoveToLane(Position.Middle);
    }
  } else if (moveRight) {
    if (position == Position.Middle) {
      MoveToLane(Position.Right);
    } else if (position == Position.Left) {
      MoveToLane(Position.Middle);
    }
  }
  timeSinceLastLaneMove = 0f;
} else if (suki.RangeExists("Joystick")) {
  // range value as a joystick to move left and right
  float range = suki.GetRange("Joystick");
  // map 0 to 1 -> -1 to +1 with a deadzone (%) in the center
  float xPercent = Suki.Utilities.Map(range, -1, 1, Deadzone);
  // apply the left/right force to the ship
  float hForce = -xPercent * MoveFactor;
  ship.AddForce(new Vector3(hForce, 0f, 0f));
}
```

Listing 5.1: Sample *Citadel* movement code excerpt referencing SUKI inputs

With the above code in place, the developer has created four named *input anchors* (i.e., “Placement”, “MoveLeft”, “MoveRight”, and “Joystick”) that, with the “MoveLeft” and “MoveRight” combined, correspond to the three *movement modes* discussed above. These anchors can be referenced by a SUKI schema to provide specific information for how that input will be collected and responded to within the game. In this way, the input is abstracted and allowed to be defined by the external schema without any need to recompile the software.

As a best practice, the developer also includes a constraint input, so that users can optionally specify a movement or posture requirement the player must satisfy during gameplay. If the player’s positioning in any frame does not satisfy the constraint, the game should implement a consequence with feedback (e.g., a UI or gameplay change). In this case, the developer changes the ship’s color and disables movement of the ship if the constraint signal exists and is not active, placing the code from Listing 5.2

before the input interpretation code to exit the function early.

```
// only continue if defined constraints are met
if (suki.SignalExists("Constraint")) {
  if (!suki.GetSignal("Constraint")) {
    // change the ship's color to indicate error
    shipRenderer.material.SetColor("_EmissionColor", error);
    return;
  }
  // change the ship's color back to normal
  shipRenderer.material.SetColor("_EmissionColor", normal);
}
```

Listing 5.2: Sample *Citadel* constraint code referencing SUKI inputs

With the SUKI library included in the project and these code statements in place, the developers can compile the software and begin constructing SUKI schemas that target the abstract input anchors provisioned in their code. As a best practice, the developer should also include documentation of these input anchors for end-users who wish to construct custom schemas.

5.1.2 Schema Construction for Cerebral Palsy Therapy

To effectively deploy *Citadel* in therapy sessions with patients with CP, we consulted CP rehabilitation experts at the Drexel College of Nursing and Health Professionals (CNHP) for movements frequently prescribed in their therapy treatments for upper body motility. As with many neuromuscular conditions, the reported therapy treatments often target Activities of Daily Living (ADL). ADL behaviors are a set of “meaningful activities that incorporate specific functional movements” [1] that reflect those a person would be expected to perform routinely in the course of self-care, including grooming, bathing, self-feeding, and so on.

The traditional set of ADL behaviors [47] along with the broader Instrumental

Activities of Daily Living (IADL) [69] are often referenced as named tasks and subtasks such as *Bring Oil to Pan*, *Bring Second Slice to Toaster*, and *Put Pan in Sink*. Because many of these routine movements rely heavily on the patient's shoulder flexion and range of motion, arm movement is a prime consideration for many therapy routines [1].

To implement these movements as input into *Citadel*, we created three sets of schemas to respectively address each of our three movement modes provisioned within the game. A therapist would not deploy all of these schema sets at once but rather would choose among these options depending on the intended exercise and therapy goals.

```
{
  "name": "Placement",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "RightShoulder", "RightHand" ],
  "reduction": { "operator": "XValue" },
  "bounds": { "extents": true }
}
```

Listing 5.3: SUKI schema for driving the "Placement" input via shoulder abduction

The first schema (Listing 5.3) reads the player's movement for the first exercise pattern (shoulder adduction and abduction) in the right shoulder to drive the the "placement" input mode provisioned within the game. It does so by reading the horizontal distance between the right hand and the right shoulder (relative to the sensor's camera space) to render a range value. Using extents to track the furthest distance achieved by the player, this schema will render a number between zero and one conveying the distance the player is currently reaching to the right bounded between their right shoulder and that extent. Extending the arm straight out will move the ship to the right lane, lowering the arm down will move the ship to the left lane, and holding the arm out at a central angle will place the ship in the center lane.

The supervision of a therapist could ensure that the patient is maintaining proper form and secondary movement objectives while engaging the motion. Optionally, SUKI could also be configured to enforce that form within the game via the constraint signal. For example, a separate schema could be set to monitor the upright posture of the player and fire the constraint signal if the player leans too far while extending, as illustrated previously in Fig. 4.10.

The above scenario could also be implemented as an AngleBetween schema that monitored the angle formed in the node triad of the RightHand, RightShoulder, and HipCenter nodes. This alternative schema could also pair with a constraint schema that ensured the normal vector of that triad was within a threshold of the forward vector $\hat{k}$ using a cross product calculation and a dot product reduction, thereby only registering the player's arm movement when performed in the proper physiological plane.

The second schema set (Listings 5.4 and 5.5) reads the player's movement for the second exercise pattern (shoulder flexion and extension) on both the left and right arms to respectively drive the "MoveLeft" and "MoveRight" inputs within the game. They do so by reading the vertical distance between the hand and the shoulder, the maximum of which the player can achieve only by raising the hand high over the head.

```
{
  "name": "MoveLeft",
  "resolution": "Signal",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "LeftShoulder", "LeftHand" ],
  "reduction": { "operator": "YValue" },
  "condition": { "operator": "GreaterThan", "percentage": 0.8 }
}
```

Listing 5.4: SUKI schema for driving the "MoveLeft" input via left shoulder flexion and extension

```
{
  "name": "MoveRight",
  "resolution": "Signal",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "RightShoulder", "RightHand" ],
  "reduction": { "operator": "YValue" },
  "condition": { "operator": "GreaterThan", "percentage": 0.8 }
}
```

Listing 5.5: SUKI schema for driving the "MoveRight" input via right shoulder flexion and extension

Once again, therapist oversight can ensure proper form (e.g., perhaps preferring a stretch of the arm forward and up rather than to the side), many aspects of which an optional constraint schema could also enforce within the game. Note that this scenario is an example of multiple schemas deployed simultaneously, creating a compound metric; by loading multiple schemas, separate kinematic data relating to the movement of the right and left arms can be interpreted independently to drive the gameplay.

This independence of input is crucial in the case of therapy, where a patient may have reduced motility in one arm compared to the other. By referencing extents (i.e., “percentage”) in the condition, these schemas will enable the players to move the ship right or left whenever they raise their respective arm above 80% of their previously demonstrated maximum for that arm. As a common consideration in some therapy scenarios, a player with reduced motility in one arm will not need to raise that arm as high as the other to achieve the same movement in the game.

The third schema (Listing 5.6) reads the player’s movement for the third movement mode (horizontal shoulder adduction and abduction) in the left arm to target the “Joystick” input anchor. It does so by examining the horizontal distance between the left hand and the left shoulder of the player. As the player moves the left outstretched arm sideways across and in front of the body, this will render a series of values that will normalize to a range between zero and one. Placing the arm near the center of that range will keep the ship stationary while moving it left or right will guide the ship left and right at a speed corresponding to the distance of the player’s hand from that center point of the demonstrated capability range.

```
{
  "name": "Joystick",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "LeftShoulder", "LeftHand" ],
  "reduction": { "operator": "XValue" },
  "bounds": { "extents": true }
}
```

Listing 5.6: SUKI schema for driving the “Joystick” input via horizontal shoulder adduction and abduction

Note that this schema is similar to the schema presented earlier in Listing 5.3, where it similarly could be replaced with an AngleBetween schema that monitored the angle formed between the chest and the left arm. Similarly, this schema could pair with a constraint that ensured the normal of the node triad was within a threshold of the upward vector $\hat{\jmath}$. Optimally, therapist oversight would also ensure proper ancillary form requirements while the game provided the motivation to engage in the therapy.

For convenience, we constructed a simple UI in the game to allow loading and unloading schema files from disk. Using this UI, a therapist can add or remove these three scenario sets on demand from within the game, allowing a patient to engage in multiple therapy exercises within a single application launch. Following our addition of these CP therapy schemas, *Citadel* is currently undergoing research within Drexel CNHP for its potential efficacy as a CP rehabilitation therapy tool.

In this case study, we examined SUKI's ability to provide a system for abstracted input via named anchors in the game's input routines, which allowed for updating the mapping between player body movement and gameplay at runtime without a need to recompile the game binary. We demonstrated the use of schemas to target players seeking different exercises for therapy and the use of extents to individualize that therapy for varying patient capabilities. We confirmed support for multiple schemas operating at once and explored the steps necessary to integrate SUKI when constructing a therapy AVG.

5.2 Case Study 2: Re-purposing a Therapy AVG Provisioned with SUKI

After completing and deploying *Citadel* for patient testing with our consultant CP therapy rehabilitation specialists, we again consulted with therapists at the University of Pennsylvania who specialized in rehabilitation for another disability with a different patient population – namely Parkinson's disease (PD). In this second case study, we examine the potential and suitability for adapting AVGs provisioned with the SUKI library toward multiple therapeutic goals and patient needs.

5.2.1 Schema Construction for Parkinson's Disease Therapy

Patients affected by PD present with an array of symptoms, including tremor, slowed movement (bradykinesia), rigidity, stooped posture (camptocormia), delayed postural responses (resulting in an increased risk of falling), cognitive impairment, and others [23, 41]. Like CP, a frequent treatment for PD is physical rehabilitation therapy, which does not aim to affect the process of the disability directly but rather improve function by encouraging compensatory activities that promote increased strength and control in the same aspects that have been diminished by the disability [48].

Though its distinctive tremor is perhaps the most well-known symptom of PD, camptocormia is often the most prominent symptom at the time of diagnosis [23]. Because a person's stance can contribute to their overall posture, it is debated whether the balance issues resulting from poor postural responses are an epiphenomenon that manifests as a consequence of the typical parkinsonian posture [7] (cf. Jacobs et al., 2005 [40]). Nonetheless, posture, balance, and reaching exercises are considered core areas for physical therapy in PD treatments [48].

Under the direction of our experts, we implemented a set of schemas that mapped the movements from several PD exercise routines toward *Citadel's* gameplay objectives. With the camptocormia condition in mind, of particular importance was a requirement that the patients perform the movements with a straightened, upright posture. Therefore, we created a constraint schema (Listing 5.7) that would ensure the patients do not curve their backs forward as they may be naturally prone to do. The schema implements the constraint by measuring the z-axis component of the position of the player's head relative to that of their pelvis, verifying that the difference does not exceed a reasonable threshold in the forward direction. In our case, we are not concerned with side leaning, and it may even be encouraged in some routines.

```
{
  "name": "Constraint",
  "resolution": "Signal",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "HipCenter", "Head" ],
  "reduction": { "operator": "ZValue" },
  "condition": { "operator": "LessThan", "scalar": 0.2 }
}
```

Listing 5.7: SUKI schema for a straight posture constraint

With the compensatory posture component in place, our next task was to map additional therapy exercise movements to the named anchors provisioned within the game. Like therapy for CP patients, PD physical therapy often targets ADL function, which rehabilitation programs that combine activity-related exercises have been shown to improve [48].

Many patients with PD exhibit abnormal postural responses in the lower body to regulate balance, which interrupts the patient's ability to respond effectively to a center of gravity displacement [7]. Therefore, exercises were presented that focus on balance during changes in center of gravity. We created the schemas in Listings 5.8 and 5.9 to implement these movements as gameplay input in *Citadel*.

```
{
  "name": "Placement",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "LeftFoot", "HipCenter" ],
  "reduction": { "operator": "XValue" },
  "bounds": { "extents": true }
}
```

Listing 5.8: SUKI schema for hip shifting

Under the supervision of the therapist, the patient stands with feet planted at shoulder width. While maintaining a straight back, the patient is then encouraged to shift his or her hips left and right without moving either foot. With a focus on hip movement, the head and torso may even remain stationary but are allowed to translate with the player's hips if desired. SUKI will monitor the difference between the horizontal location of the player's foot and the center of the pelvis, converting the current value within the maximum demonstrated extents to a range value for the "Placement" anchor. With the player's center of gravity balanced between the feet, the ship will move to or remain in the center lane, where shifting the center of gravity toward the left or right foot will move the ship to the corresponding lane.

```
{
  "name": "Joystick",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "HipCenter", "Head" ],
  "reduction": { "operator": "XValue" },
  "bounds": { "extents": true }
}
```

Listing 5.9: SUKI schema for leaning torso

In the leaning torso scenario presented in Listing 5.9, the range value is derived from the difference in horizontal position between the player's head and pelvis center. As the player leans sideways left and right, an extent range will develop within which the patient can guide the ship using his or her upper body as a virtual joystick. For variety in gameplay, the "Joystick" input anchor is targeted to allow the player to move freely among the lanes.

Additional PD exercises employed by our experts focus on the opposed objectives of extending reach and maintaining balance. One such exercise is a cross-over reach, where players are encouraged to stretch an arm forward across the body and out to the opposite side of that arm's shoulder. We created the schemas in Listings 5.10 and 5.11 to implement this movement for both arms.

```json
{
  "name": "MoveRight",
  "resolution": "Signal",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "HipCenter", "LeftHand" ],
  "reduction": { "operator": "XValue" },
  "condition": { "operator": "GreaterThan", "percentage": 0.8 }
}
```

Listing 5.10: SUKI schema for left arm crossover

```json
{
  "name": "MoveLeft",
  "resolution": "Signal",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "HipCenter", "RightHand" ],
  "reduction": { "operator": "XValue" },
  "condition": { "operator": "GreaterThan", "percentage": 0.8,
     "reverse": true }
}
```

Listing 5.11: SUKI schema for right arm crossover

While keeping an upright posture via the constraint in Listing 5.7, the player is encouraged to reach the left arm across the body and out to the right to move the ship right. Similarly, reaching the right arm across the body and out to the left will move the ship left. Under the supervision of the therapist, twisting of the upper body is also encouraged to add additional balance consideration and maximize reach distance. In this way, the schemas differ from the approach in the CP scenario depicted in Listing 5.6, where here the distance is measured from a central location within the body instead of the shoulder. Where the CP exercise intended to isolate horizontal abduction and adduction of the shoulder, these PD scenarios are meant to actively promote leaning in the upper body.

In both cases, the corresponding signal input triggers only when the player achieves a percentage of the previously observed extent. Note that the schema monitoring the right arm uses the "reverse" flag to mirror the calculations for the left side, similar to the schema presented earlier in Figure 4.8.

As with the CP therapy schemas, these three scenario sets can be added or removed on demand from within the game, allowing a patient to engage in multiple exercise routines without restarting the application. Following our addition of these schemas, PD specialists at the University of Pennsylvania are now also conducting research with *Citadel* to assess its potential efficacy as a PD rehabilitation therapy tool.

In this case study, we explored the potential for SUKI to enable a single application to target multiple disabilities and therapeutic goals. The next test case examines the process of adding SUKI to an existing therapy game and the potential it may have to improve or extend the functionality of that game.

5.3 Case Study 3: Integrating SUKI into an Existing Therapy AVG

After implementing a game with the SUKI system and demonstrating its ability to adapt an AVG's therapeutic purpose without any change to the game itself, we then set out to integrate SUKI into an existing AVG constructed with traditional input routines based on direct kinematic sensor data. In doing so, we explore both its ease of integration and its potential for extending existing therapy games.

Kollect is a Kinect-based AVG therapy game for adolescents with CP designed to promote health and functional mobility in arms and legs. Though its objectives

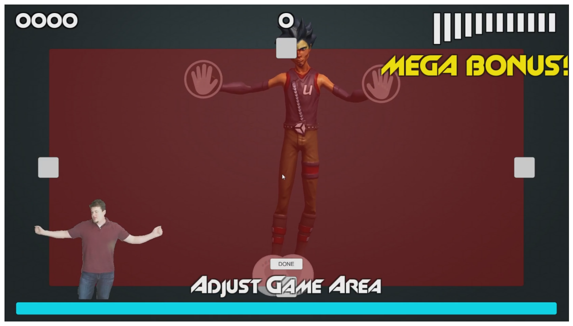


Figure 5.3: In *Kollect*, players are represented as avatars on the screen that mirror their movements in real life. Markers attached to the avatar hands and feet will interact with items on screen.

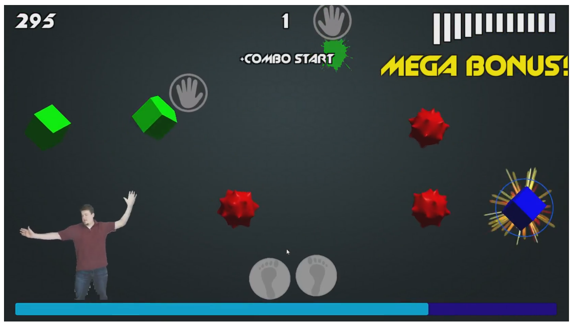


Figure 5.4: Over several rounds, a set of objects will spawn that players will attempt to collect for points. Players move their arms and legs to collect the target objects while avoiding the red hazards.

do promote strength and movement planning, the primary metric for the game is patient heart rate, encouraging a more active and frenetic play style over a methodical approach.

In *Kollect*, an avatar representing the patient stands or sits in the center of the screen, where movements of the avatar directly mirror the movement of the player as detected by a Kinect sensor. Within the game, floating markers attach to the locations of the avatar's hands and feet; though the markers are locked in the xy-plane within the game, they can move around the screen as a result of the player moving his or her hands and feet and driving the same movement in the avatar representation. Each round, a set of objects will spawn and float around the avatar. The player can "collect" these objects by making contact with them via the markers. In some game modes, hazard objects also spawn as a way to generate an additional challenge and encourage more careful movement planning; the player must attempt to collect all of the target objects while avoiding the hazard objects to maximize the resulting final score (Figures 5.3 and 5.4).

One limitation of *Kollect's* original input routines rests in the fact that the markers adhere to the avatar's hands and feet in the center of the screen, restricting the game from taking advantage of the full horizontal screen space. Additionally, if the patient has asymmetric capabilities for flexion and reach, it is possible to spawn objects that may not be reachable on a particular side. *Kollect* currently manages both concerns with a manual calibration square defined before the game starts, in which the user designates the areas in which objects may spawn. However, with an implementation using SUKI, this manual calibration may no longer be necessary.

5.3.1 Code Integration

Integrating SUKI into *Kollect* involved adding the library to the project and updating the input code in the main game loop. Written in the Unity game engine, *Kollect* manages the hand and foot marker locations using a custom game object with an input script and four child objects representing the four markers within the scene. Our programming task was to update this input script to manipulate the position of the four markers according to SUKI input data, rather than avatar body position derived directly from the Kinect sensor.

With the world space origin near the center of the scene camera, the four markers usually cover separate zones within the screen. For example, the right arm is intended to be used to collect objects that appear in the upper right quadrant, and the left foot is meant to be used to collect objects that spawn in the lower left quadrant. Therefore, we decided to bound the movement of the markers to specific areas of the screen to enforce this play style. This design was accomplished by pre-calculating screen rectangles for each zone in the script's Awake() function, as shown in Listing 5.12.

```
// get the rect of the screen in world space
Vector3 botLeft = Camera.main.ViewportToWorldPoint(new
   Vector3(0, 0, mainCam.nearClipPlane));
Vector3 topRight = Camera.main.ViewportToWorldPoint(new
   Vector3(1, 1, mainCam.nearClipPlane));
// determine the widths and heights of the active zones
float width = topRight.x;
float hHeight = topRight.y;
float fHeight = -botLeft.y;

// assign the rectangles for active zones of the markers
rects = new Dictionary<Transform, Rect>();
// Rect is (x,y,w,h) with (0,0) in bottom left corner
rects.Add(lh, new Rect(botLeft.x, 0f, width, hHeight));
rects.Add(rh, new Rect(0f, 0f, width, hHeight));
rects.Add(lf, new Rect(botLeft.x, botLeft.y, width, fHeight));
rects.Add(rf, new Rect(0f, botLeft.y, width, fHeight));
// add a rectangle that encompasses the entire screen
rects.Add(this.transform,
new Rect(0f, 0f, 2 * width, hHeight + fHeight));
```

Listing 5.12: Sample *Kollect* code excerpt for screen segmenting

With definitions for the four zones, our Update() function establishes four Location2D input anchors to move each marker within its respective zone. Additionally, as a best practice, we include a check for a constraint with a UI cue that changes the colors of the hand and foot markers so that a therapist may optionally add one. With a set of class variables, constants, and references to the four child objects (via their Transform components), the input code for *Kollect* using SUKI inputs is shown in Listing 5.13.

```
// check for constraint
if (suki.SignalExists("Constraint")) {
  if (!suki.GetSignal("Constraint")) {
    ChangeMarkerColors(error);
    return;
  }
  ChangeMarkerColors(normal);
}
...
// mode 1: move each marker within its respective zone
if (suki.Location2DExists("LeftHand")) {
  rect = rects[lh];
  pos = suki.GetLocation2D("LeftHand");
  newPos.x = -pos.x * (rect.width) + rect.xMax;
  newPos.y = pos.y * (rect.height) + rect.yMin;
  lh.position = Vector3.Lerp(lh.position, newPos, 1f);
}
if (suki.Location2DExists("RightHand")) {
  rect = rects[rh];
  pos = suki.GetLocation2D("RightHand");
  newPos.x = pos.x * (rect.width) + rect.xMin;
  newPos.y = pos.y * (rect.height) + rect.yMin;
  rh.position = Vector3.Lerp(rh.position, newPos, 1f);
}
if (suki.Location2DExists("LeftFoot")) {
  rect = rects[lf];
  pos = suki.GetLocation2D("LeftFoot");
  newPos.x = -pos.x * (rect.width) + rect.xMax;
  newPos.y = pos.y * (rect.height) + rect.yMin;
  lf.position = Vector3.Lerp(lf.position, newPos, 1f);
}
if (suki.Location2DExists("RightFoot")) {
  rect = rects[rf];
  pos = suki.GetLocation2D("RightFoot");
  newPos.x = pos.x * (rect.width) + rect.xMin;
  newPos.y = pos.y * (rect.height) + rect.yMin;
  rf.position = Vector3.Lerp(rf.position, newPos, 1f);
}
```

Listing 5.13: Sample *Kollect* movement code excerpt referencing SUKI inputs

With schemas to drive the named Location2D inputs based on the player's hands and feet in real world space, the above implements the movement objectives originally presented in the game. However, to take advantage of SUKI's flexibility, we implement a few more movement modes based on additional input anchors following this code, as shown in Listing 5.14.

```
// mode 2: single marker with coverage for the entire screen
if (suki.Location2DExists("Joystick")) {
  rect = rects[this.transform];
  pos = suki.GetLocation2D("Joystick");
  newPos.x = pos.x * (rect.width) + rect.xMin;
  newPos.y = pos.y * (rect.height) + rect.yMin;
  rh.position = Vector3.Lerp(rh.position, newPos, 1f);
}
// mode 3: single marker driven by two range inputs
if (suki.RangeExists("JoystickX")) {
  range = suki.GetRange("JoystickX");
  // map 0 to 1 -> -1 to +1 with a deadzone (%) in the center
  float percent = Suki.Utilities.Map(range, -1, 1, Deadzone);
  newPos.x += percent * MoveFactor;
  rh.position = Vector3.Lerp(rh.position, newPos, 1f);
}
if (suki.RangeExists("JoystickY")) {
  range = suki.GetRange("JoystickY");
  // map 0 to 1 -> -1 to +1 with a deadzone (%) in the center
  float percent = Suki.Utilities.Map(range, -1, 1, Deadzone);
  newPos.y += percent * MoveFactor;
  rh.position = Vector3.Lerp(rh.position, newPos, 1f);
}
```

Listing 5.14: Code excerpt for additional *Kollect* movement modes

In this way, we have set up three modes for movement. In the first set, we replicate the original gameplay by using four separate input anchors that enable each hand and foot icon to follow the player's movement of the corresponding body part.

In the second, a single Location2D input will act like a joystick, driving relative movement of a single marker that is navigated around the screen to collect objects. In the third, two range inputs also guide a single marker but isolate the magnitude of the movement between the horizontal and vertical axes. To support different therapy scenarios, an existing game parameter available in the settings determines which (if any) of the markers are active. In either of the last two input modes, the therapist should disable the unused markers for the left hand and both feet.

5.3.2 Schema Design to Replicate Existing Gameplay

Our first provisioned input mode requires four schemas to drive the Location2D inputs "LeftHand," "RightHand," "LeftFoot," and "RightFoot" based on the location of the respective body part of the player in real life. To do so, we make use of a Position node metric that tracks the player's hands and feet, as demonstrated in Listings 5.15 and 5.16.

For brevity, only schemas for the right hand and left foot are shown, but similar schemas were also created to map input from the left hand and right foot as well. These four schemas together recreate the original input configuration for the game with a few improvements (Figs 5.5 and 5.6). First, because SUKI is considering the *relative* position of each body part within its observed extents and mapping them to an area of the screen, the entirety of the display can be used for gameplay. Second, because the extents for each body part are independent, the game will adapt to each body part according to the patient's capability. With these two improvements, the manual calibration square is no longer required, but may still be used if the therapist desires to control the locations of object spawns.

```
{
  "name": "RightHand",
  "resolution": "Location2D",
  "device": "Kinect",
  "metric": "Position",
  "nodes": [ "RightHand" ],
  "bounds": { "extents": true }
}
```

Listing 5.15: SUKI schema for *Kollect* right hand motion control

```
{
  "name": "LeftFoot",
  "resolution": "Location2D",
  "device": "Kinect",
  "metric": "Position",
  "nodes": [ "LeftFoot" ],
  "bounds": { "extents": true, "reverse": true }
}
```

Listing 5.16: SUKI schema for *Kollect* left foot motion control

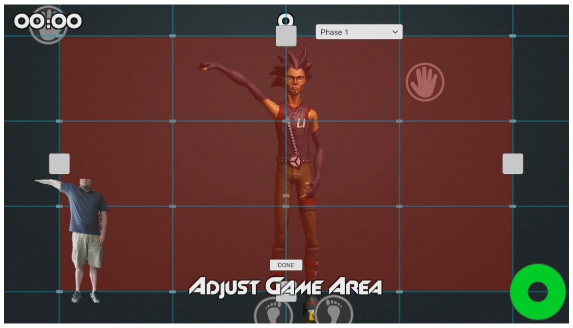


Figure 5.5: In the SUKI implementation of *Kollect's* controls, the player's hand and foot movements within their extents will map to the full size of the corresponding quadrant on screen and are no longer bound to the avatar.

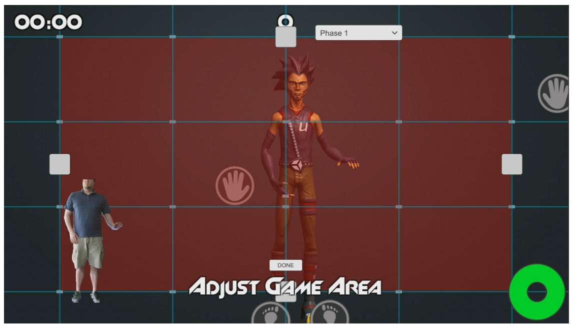


Figure 5.6: A patient with asymmetric arm movement capability will leverage independent extents. Note the mirrored right arm's limited horizontal extension moves the corresponding hand marker all the way to the edge of the screen.

5.3.3 Leveraging SUKI for Alternate Play Modes

With SUKI now implemented for the traditional *Kollect* gameplay, we take advantage of the additional input anchors to explore alternative play modes. In this example, the new exercises have not been designed to target a particular condition under the guidance of a rehabilitation expert but were constructed as a thought experiment to demonstrate the opportunities available in a game built with SUKI.

Where the current primary objective of *Kollect* is to increase movement and cardio output, there may be an opportunity to design a mode that focuses more on movement planning. For this mode, demonstrated in Listing 5.17, we target the "Joystick" input anchor that reads the player's right hand as a "virtual mouse" to guide a single marker around the screen. Moving the right hand up and down will move the marker up and down at a speed relative to the current vertical position between the player's demonstrated extents. Similarly, moving the right hand laterally in front of the body will move the marker sideways on the screen. In this mode, the player should attempt to carefully guide the marker to pick up target objects while remaining careful to avoid hazards.

```json
{
  "name": "Joystick",
  "resolution": "Location2D",
  "device": "Kinect",
  "metric": "Position",
  "nodes": [ "RightHand" ],
  "bounds": { "extents": true }
}
```

Listing 5.17: Schema for driving the "Joystick" input from right hand movement

As a second alternative movement mode, we target the isolated "JoystickX" and "JoystickY" input anchors, where the magnitudes derive from independent range inputs. Considering our previous exploration into balance exercises, we designed a set of schemas that would enable players to use their upper bodies as virtual joysticks to guide the single marker on the screen. In this scenario, implemented in Listings 5.18 and Listing 5.19, leaning the upper body forward and backward at the hips would guide the marker up and down, and tilting side to side at the hips would guide the marker left and right.

```
{
  "name": "JoystickX",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "HipCenter", "Head" ],
  "reduction": { "operator": "XValue" },
  "bounds": { "extents": true }
}
```

Listing 5.18: Schema for driving horizontal marker movement via sideways lean

```
{
  "name": "JoystickY",
  "resolution": "Range",
  "device": "Kinect",
  "metric": "VectorBetween",
  "nodes": [ "HipCenter", "Head" ],
  "reduction": { "operator": "ZValue" }
  "bounds": { "extents": true }
}
```

Listing 5.19: Schema for driving vertical marker movement via forward lean

Whether or not these movement scenarios would be useful in AVG therapy for a particular patient and disability would be for a therapist to determine, but these examples serve to demonstrate the ability for SUKI to extend the potential of an

AVG to support more varied exercise opportunities without recompilation than those created with traditional programming methods for body motion input. Additionally, these schemas serve as an example of what end-user therapists would be free to create on their own to suit their needs for a particular patient or exercise routine. As demonstrated, this functionality enables the therapist to modify a very fundamental aspect of the game without requiring any modification to the software.

In this case study, we have examined another aspect of SUKI's potential ease of integration – namely, the steps necessary to integrate it into a completed game to both replicate the existing input functionality and extend it to support other modes of play. In doing so, we have examined SUKI's suitability as a modular library independent from its initial game and confirmed its requirements in a separate environment.

6. CONCLUSION AND FUTURE WORK

The focus of our research was to design a flexible and adaptive system for interpreting human body input to ease development and enhance the capability of AVGs for therapy. In doing so, we aimed to examine a system that could leverage player capability as a basis for dynamic adaptation. In pursuit of this goal, we developed the *System for Unified Kinematic Input* (SUKI), a software library that provides an abstraction for input routines within game environments and an adaptation system that can individualize gameplay. With oversight from therapy researchers for multiple neuromuscular conditions, we examined the capability for SUKI to create flexible input configurations for games and adjust its measurements to patients without a need for manual calibration.

Just as the sophistication of human motion input devices has enabled more advanced integration between player motion and AVG gameplay, we believe that leveraging more intelligent game systems will unlock future enhancement to the capability, usefulness, and ultimately the efficacy of AVG-based therapies. However, work remains both to extend this investigation and validate its effect in real-world scenarios.

6.1 Limitations and Future Work

The current implementation of SUKI extents is fairly simple and serves as a demonstration of the concept. Though it implements a player model, SUKI extents currently only leverage a single dimension for each movement. More complex extents and more sophisticated artificial intelligence techniques (such as machine learning) might be incorporated to more effectively assess player movement limitations and provide a more sophisticated basis for adaptation.

Concerning meeting the individual therapy needs of the user, SUKI can make AVGs more adaptable, but not necessarily as *adaptive* as they could be. Because the extents measured for a player are only applicable within the context of the particular schema, the system can only adapt with respect to that schema. Moving forward, a more sophisticated system could derive an expectation of movement built from knowledge of the physiology and condition of the player rather than demonstrated ability. Additionally, profiles based on body measurement could aim to predict capability in a given motion, rather than assessing movement limits empirically.

Though JSON enables flexibility and expressiveness in creating SUKI schemas, a more accessible end user solution may benefit from a robust UI for crafting schemas. A future version of SUKI may be able to construct a schema automatically after watching the player or therapist demonstrate the movement they desire to perform in the therapy. Additionally, extents measured in one schema may be applicable to others, especially if they are built on a similar motion; carrying over knowledge across schemas might help the system better anticipate player capability in multiple scenarios.

The SUKI architecture and schema definition were designed to support multiple kinematic devices, and the initial version of SUKI implements support for the Microsoft Kinect (v2) and Orbbec Astra sensors. Future work could expand this to support additional sensors with the necessary SDK integration that would grant an even more device-agnostic view of player kinematic data. Additionally, the current SUKI design offers a curated set of node metrics, input resolutions, and calculation and reduction operators described in Section 4.3.4. This design is also open to extension as developers explore more complex movements, find need for additional input anchor data types, or desire more pre-processing options for kinematic sensor data.

This research presents a software system that aids in the design of motion-based video games, but it emphasizes an application in AVG therapy experiences. Future work, including studies that our clinical partners are currently conducting with promising preliminary results, should evaluate SUKI-enabled games for their therapeutic efficacy. These studies can also examine whether adaptive input can improve long-term therapy metrics, such as retention, adherence, and patient outcomes.

Finally, beyond the therapy application emphasized in this work, SUKI should be evaluated for its potential benefits to non-therapy AVGs and broader player populations. Studies might explore player attitudes toward capability-based dynamic games and user engagement in entertainment AVGs built with SUKI. This may also include developer studies that examine the proposed improvements to software flexibility and reduced burden of development.

6.2 Implications and Application

Beyond the proposed tangible benefits of easier AVG development and extended breadth of usability that SUKI can provide a therapy AVG, the most exciting benefit is that of using player capability as a basis for dynamic adaptation. Further exploration can be pursued among multiple paths from this research.

Beyond Body Movement. SUKI currently focuses only on musculoskeletal physical ability, such as movement planning and range of motion. Other aspects of player physical capability, such as visual acuity, reflex time, or dexterity might provide more interesting or appropriate bases for adaptation, depending on the nature of the game it is augmenting. For example, a console first-person shooter could intelligently adapt to a player's thumb dexterity by automatically increasing or decreasing dead zone and sensitivity on the analog joystick, dynamically adjusting the degree of aim assistance, or modifying the timescale of the environment.

Beyond Physical Traits. Not limited to physical capability, other aspects of individual difference among players (e.g., cognitive, temperamental, lifestyle) might serve as viable bases for adaptation. These might include qualities of the player such as memory, spatial reasoning, patience, or even current fatigue level. For example, a puzzle game might consider a player's working memory (WM), such as their score in a WM span test [18, 46], to provide more intensive scaffolding or to inform its hint system. Similarly, an action adventure game with DDA capabilities could modify its difficulty curve based on its understanding of the player's temperament or tolerances (e.g., patience and persistence).

Beyond Input. SUKI in its current form offers dynamic adaptation in the way a game processes and interprets input. Though of particular importance in the case of AVGs, this limits the potential scope of effect of the dynamic adaption to the portions of the game experience that address direct player interaction. This is the most natural place to begin exploring capability-based adaptation, but future work may explore the potential to adjust elements of the gameplay mechanics or virtual environment.

Bibliography

[1] R. J. Adams, M. D. Lichter, E. T. Krepkovich, A. Ellington, M. White, and P. T. Diamond. Assessing upper extremity motor function in practice of virtual activities of daily living. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(2):287–296, 2015.

[2] J. T. Alexander, J. Sear, and A. Oikonomou. An investigation of the effects of game difficulty on player enjoyment. *Entertainment Computing*, 4(1):53–62, 2013.

[3] A. A. AlSaif and S. Alsenany. Effects of interactive games on motor performance in children with spastic cerebral palsy. *J Phys Ther Sci*, 27(6):2001–3, 2015.

[4] S. C. J. Bakkes, P. H. M. Spronck, and H. Jaap van den Herik. Opponent modelling for case-based adaptive game AI. *Entertainment Computing*, 1(1): 27–37, 2009.

[5] A. Baldwin, D. Johnson, and P. A. Wyeth. The effect of multiplayer dynamic difficulty adjustment on the player experience of video games. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems*, pages 1489–1494. ACM, 2014.

[6] R. Bartle. Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD research*, 1(1):19, 1996.

[7] B. R. Bloem, D. J. Beckley, and J. G. van Dijk. Are automatic postural responses in patients with Parkinson's disease abnormal due to their stooped posture? *Experimental brain research*, 124(4):481–488, 1999.

[8] I. Bogost. The rhetoric of exergaming. In *Proceedings of the Digital Arts and Cultures (DAC)*, 2005.

[9] B. Bonnechere, B. Jansen, L. Omelina, M. Degelaen, V. Wermenbol, M. Rooze, and S. Van Sint Jan. Can serious games be incorporated with conventional treatment of children with cerebral palsy? A review. *Res Dev Disabil*, 35(8): 1899–913, 2014.

[10] K. J. Bower, R. A. Clark, J. L. McGinley, C. L. Martin, and K. J. Miller. Clinical feasibility of the nintendo wii for balance training post-stroke: a phase

II randomized controlled trial in an inpatient setting. *Clinical rehabilitation*, 28 (9):912–923, 2014.

[11] G. Branwyn and G. Clabaugh. Computerscan: Working out with your computer. *The Futurist*, 23(1), Jan 1989.

[12] M. W. Brault, U. S. Economics, and S. A. U. C. Bureau. *Americans with disabilities, 2010: Household economic studies*. Current population reports. U.S. Dept. of Commerce, Economics and Statistics Administration, U.S. Census Bureau, Washington, D.C., 2012.

[13] D. Charles and M. Black. Dynamic player modeling: A framework for player-centered digital games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, 2004.

[14] D. Charles, A. Kerr, M. McNeill, M. McAlister, M. Black, J. Kcklich, A. Moore, and K. Stringer. Player-centred game design: Player modelling and adaptive digital games. In *Proceedings of the Digital Games Research Conference*, volume 285, page 00100, 2005.

[15] J. Chen. *Flow in games*. Master's thesis, Univ. Southern California, 2006.

[16] N. R. Cherabuddi. Exergaming: Video games as a form of exercise. *Brown University. Department of Computer Science*, 2011.

[17] J. H. Choi, E. Y. Han, B. R. Kim, S. M. Kim, S. H. Im, S. Y. Lee, and C. W. Hyun. Effectiveness of commercial gaming-based virtual reality movement therapy on functional recovery of upper extremity in subacute stroke patients. *Ann Rehabil Med*, 38(4):485–93, 2014.

[18] A. R. A. Conway, M. J. Kane, M. F. Bunting, D. Z. Hambrick, O. Wilhelm, and R. W. Engle. Working memory span tasks: A methodological review and user's guide. *Psychonomic Bulletin & Review*, 12(5):769–786, 2005.

[19] B. Cowley and D. Charles. Behavlets: A method for practical player modelling using psychology-based player traits and domain specific features. *User Modeling and User-Adapted Interaction*, 26(2-3):257–306, 2016.

[20] M. Csikszentmihalyi. *Flow: The psychology of optimal experience*. Harper & Row, 1990.

[21] D. L. Damiano. Activity, activity, activity: Rethinking our physical therapy approach to cerebral palsy. *Phys Ther*, 86(11):1534–40, 2006.

[22] P. J. Diefenbach, R. C. Gray, T. J. Day, and M. E. O'Neil, *Patient Data Visualization, Configuration of Therapy Parameters from a Remote Device, and Dynamic Constraints*, United States, Patent: 62/333,142, 2016.

[23] R. Djaldetti, R. Mosberg-Galili, H. Sroka, D. Merims, and E. Melamed. Camptocormia (bent spine) in patients with Parkinson's disease - characterization and possible pathogenesis of an unusual phenomenon. *Movement Disorders*, 14(3): 443–447, 1999.

[24] Doctor Kinetic BV, *Doctor Kinetic: Functional Rehabilitation in Virtual Reality*, 2016. Available: `http://doctorkinetic.com`, Accessed: November 10, 2016.

[25] T. Dutta. Evaluation of the kinect sensor for 3-D kinematic measurement in the workplace. *Applied Ergonomics*, 43(4):645–649, 2012.

[26] J.-F. Esculier, J. Vaudrin, P. Bériault, K. Gagnon, and L. E. Tremblay. Home-based balance training programme using Wii Fit with balance board for Parkinson's disease: A pilot study. *Journal of Rehabilitation Medicine*, 44(2):144–150, 2012.

[27] G. M. Farouk, I. F. Moawad, and M. Aref. Generic opponent modelling approach for real time strategy games. In *Computer Engineering and Systems (ICCES) 2013 8th International Conference*, pages 21–27. IEEE, 2013.

[28] Fitness Gaming Group Ltd., *VAST Rehab: Rehabilitation with Biofeedback*, 2016. Available: `http://vast.rehab`, Accessed: November 10, 2016.

[29] A. N. Foster. *Gaming their way: Learning in simulation strategy video games?* PhD thesis, Michigan State University, 2009.

[30] I. Franki, C. Van den Broeck, J. De Cat, W. Tijhuis, G. Molenaers, G. Vanderstraeten, and K. Desloovere. A randomized, single-blind cross-over design evaluating the effectiveness of an individually defined, targeted physical therapy approach in treatment of children with cerebral palsy. *Clinical Rehabilitation*, 28 (10):1039–52, 2014.

[31] T. Fullerton. *Game design workshop: A playcentric approach to creating innovative games.* CRC press, 2014. ISBN 1482217171.

[32] B. Galna, D. Jackson, G. Schofield, R. McNaney, M. Webster, G. Barry, D. Mhiripiri, M. Balaam, P. Olivier, and L. Rochester. Retraining function in people with Parkinson's disease using the Microsoft Kinect: Game design and pilot testing. *J Neuroeng Rehabil*, 11:60, 2014.

[33] S. Gauggel and S. Fischer. The effect of goal setting on motor performance and motor learning in brain-damaged patients. *Neuropsychological Rehabilitation*, 11 (1):33–44, 2001.

[34] L. E. F. Graves, N. D. Ridgers, K. Williams, G. Stratton, G. Atkinson, and N. T. Cable. The physiological cost and enjoyment of Wii Fit in adolescents, young adults, and older adults. *Journal of Physical Activity & Health*, 7(3):393–401, 2010.

[35] R. O. Gutierrez, F. Galan Del Rio, R. Cano de la Cuerda, I. M. Alguacil Diego, R. A. Gonzalez, and J. C. Page. A telerehabilitation program by virtual reality-video games improves balance and postural control in multiple sclerosis patients. *NeuroRehabilitation*, 33(4):545–54, 2013.

[36] G. Hawkins, K. Nesbitt, and S. Brown. Dynamic difficulty balancing for cautious players and risk takers. *International Journal of Computer Games Technology*, 2012:3, 2012.

[37] C. Heeter. *Play styles and learning*, pages 826–846. IGI Global, Hershey, PA, 2009.

[38] H. Holmes, J. Wood, S. Jenkins, P. Winship, D. Lunt, S. Bostock, and K. Hill. Xbox Kinect represents high intensity exercise for adults with cystic fibrosis. *J Cyst Fibros*, 12(6):604–8, 2013.

[39] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.

[40] J. V. Jacobs, D. M. Dimitrova, J. G. Nutt, and F. B. Horak. Can stooped posture explain multidirectional postural instability in patients with Parkinson's disease? *Experimental brain research*, 166(1):78–88, 2005.

[41] J. Jankovic. Parkinson's disease: Clinical features and diagnosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 79(4):368–376, 2008.

[42] D. Jelsma, R. H. Geuze, R. Mombarg, and B. C. Smits-Engelsman. The impact of Wii Fit intervention on dynamic balance control in children with probable developmental coordination disorder and balance problems. *Hum Mov Sci*, 33: 404–18, 2014.

[43] C. Jennett, A. L. Cox, P. Cairns, S. Dhoparee, A. Epps, T. Tijs, and A. Walton. Measuring and defining the experience of immersion in games. *International Journal of Human-Computer Studies*, 66(9):641–661, 2008.

[44] Jintronix, Inc., *Jintronix: Sense your Progress*, 2016. Available: `http://www.jintronix.com/`, Accessed: November 10, 2016.

[45] L. Y. Joo, T. S. Yin, D. Xu, E. Thia, P. F. Chia, C. W. K. Kuah, and K. K. He. A feasibility study using interactive commercial off-the-shelf computer gaming in upper limb rehabilitation in patients after stroke. *Journal of rehabilitation medicine*, 42(5):437–441, 2010.

[46] M. J. Kane and R. W. Engle. The role of prefrontal cortex in working-memory capacity, executive attention, and general fluid intelligence: An individual-differences perspective. *Psychonomic Bulletin & Review*, 9(4):637–671, 2002.

[47] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, and M. W. Jaffe. Studies of illness in the aged: The index of ADL: A standardized measure of biological and psychosocial function. *Jama*, 185(12):914–919, 1963.

[48] S. H. Keus, B. R. Bloem, E. J. Hendriks, A. B. Bredero-Cohen, and M. Munneke. Evidence-based analysis of physical therapy in Parkinson's disease with recommendations for practice and research. *Movement disorders*, 22(4):451–460, 2007.

[49] K. Kiili. Digital game-based learning: Towards an experiential gaming model. *The Internet and Higher Education*, 8(1):13–24, 2005.

[50] A. Kramer, C. Dettmers, and M. Gruber. Exergaming with additional postural demands improves balance and gait in patients with multiple sclerosis as much as conventional balance training and leads to high adherence to home-based balance training. *Arch Phys Med Rehabil*, 95(10):1803–9, 2014.

[51] M. F. Levin, P. L. Weiss, and E. A. Keshner. Emergence of virtual reality as a tool for upper limb rehabilitation: Incorporation of motor control and motor learning principles. *Physical therapy*, 95(3):415, 2015.

[52] D. Liebling and M. R. Morris. Kinected browser: Depth camera interaction for the web. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, pages 105–108. ACM, 2012. ISBN 1450312098.

[53] C. Liu, P. Agrawal, N. Sarkar, and C. Shuo. Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback. *International Journal of Human-Computer Interaction*, 25(6):506–529, 2009.

[54] Microsoft Corporation, *Kinect for Windows human interface guidelines v2.0*, 2014. Available: `https://go.microsoft.com/fwlink/p/?LinkID=403900`, Accessed: February 27, 2016.

[55] M. E. Morris. Movement disorders in people with Parkinson disease: A model for physical therapy. *Physical therapy*, 80(6):578, 2000.

[56] J. H. Murray. *Inventing the medium: Principles of interaction design as a cultural practice.* Mit Press, 2011.

[57] C. O'Donovan, P. Greally, G. Canny, P. McNally, and J. Hussey. Active video games as an exercise tool for children with cystic fibrosis. *J Cyst Fibros*, 13(3): 341–6, 2014.

[58] Orbbec 3D Tech. Intl. Inc., *Orbecc - Intelligent computing for everyone everywhere*, 2017. Available: `hhttps://orbbec3d.com/`, Accessed: January 10, 2017.

[59] I. Parry, C. Carbullido, J. Kawada, A. Bagley, S. Sen, D. Greenhalgh, and T. Palmieri. Keeping up with video game technology: Objective analysis of

Xbox Kinect and PlayStation 3 Move for use in burn rehabilitation. *Burns*, 40 (5):852–9, 2014.

[60] V. M. Á. Pato and C. Delgado-Mata. Dynamic difficulty adjusting strategy for a two-player video game. *Procedia Technology*, 7:315–321, 2013.

[61] J. H. Patterson. *Avian: Game design and player metrics for player modeling in educational games*. Master's thesis, Drexel University, 2014.

[62] W. Peng, J. C. Crouse, and J.-H. Lin. Using active video games for physical activity promotion: A systematic review of the current state of research. *Health Education & Behavior*, 40(2):171–192, 2013.

[63] J. E. Pompeu, F. A. Mendes, K. G. Silva, A. M. Lobo, P. Oliveira Tde, A. P. Zomignani, and M. E. Piemonte. Effect of Nintendo Wii-based motor and cognitive training on activities of daily living in patients with Parkinson's disease: A randomised clinical trial. *Physiotherapy*, 98(3):196–204, 2012.

[64] S. Radtka, R. Hone, C. Brown, J. Mastick, M. E. Melnick, and G. A. Dowling. Feasibility of computer-based videogame therapy for children with cerebral palsy. *Games Health J*, 2(4):222–228, 2013.

[65] M. Robert, L. Ballaz, R. Hart, and M. Lemay. Exercise intensity levels in children with cerebral palsy while playing with an active video game console. *Phys Ther*, 93(8):1084–91, 2013.

[66] Y. Salem, S. J. Gropack, D. Coffin, and E. M. Godwin. Effectiveness of a low-cost virtual reality system for children with developmental delay: A preliminary randomised single-blind controlled trial. *Physiotherapy*, 98(3):189–95, 2012.

[67] G. Saposnik, R. Teasell, M. Mamdani, J. Hall, W. McIlroy, D. Cheung, K. E. Thorpe, L. G. Cohen, M. Bayley, and f. t. S. O. R. C. W. Group. Effectiveness of virtual reality using wii gaming technology in stroke rehabilitation: A pilot randomized clinical trial and proof of principle. *Stroke*, 41(7):1477–1484, 2010.

[68] M. P. Silva, V. d. N. Silva, and L. Chaimowicz. Dynamic difficulty adjustment on MOBA games. *Entertainment Computing*, 18:103–123, 2017.

[69] W. D. Spector, S. Katz, J. B. Murphy, and J. P. Fulton. The hierarchical relationship between activities of daily living and instrumental activities of daily living. *Journal of chronic diseases*, 40(6):481–489, 1987.

[70] E. A. Suma, B. Lange, A. S. Rizzo, D. M. Krum, and M. Bolas. FAAST: The flexible action and articulated skeleton toolkit. In *Virtual Reality Conference (VR), 2011 IEEE*, pages 247–248. IEEE, 2011. ISBN 1457700387.

[71] S.-W. Um, T.-Y. Kim, and J.-S. Choi. Dynamic difficulty controlling game system. *IEEE transactions on consumer electronics*, 53(2):812–818, 2007.

[72] J. Valls-Vargas, S. Ontañón, and J. Zhu. Exploring player trace segmentation for dynamic play style prediction. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.

[73] W. M. van den Hoogen, W. A. IJsselsteijn, and Y. A. de Kort. Exploring behavioral expressions of player experience in digital games. In *Proceedings of the workshop on Facial and Bodily Expression for Control and Adaptation of Games ECAG*, volume 2008, pages 11–19, 2008.

[74] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the accuracy and robustness of the Leap Motion Controller. *Sensors (Basel)*, 13(5):6380–93, 2013.

[75] Mark Wilson, *Exclusive: Microsoft has stopped manufacturing the Kinect*, 2017. Available: `https://www.fastcodesign.com/90147868/exclusive-microsoft-has-stopped-manufacturing-the-kinect`, Accessed: October 25, 2017.

[76] N. Yee. The demographics, motivations, and derived experiences of users of massively multi-user online graphical environments. *Presence: Teleoperators and virtual environments*, 15(3):309–329, 2006.

[77] N. Yee. The labor of fun how video games blur the boundaries of work and play. *Games and Culture*, 1(1):68–71, 2006.

[78] Z. Zhengyou. Microsoft Kinect sensor and its effect. *MultiMedia, IEEE*, 19(2): 4–10, 2012.